LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-665543

COE Task Order 4: Final Report Automated Threat Recognition

A. P. Sales, S. Azevedo, H. Kim, H. Martz

December 22, 2014

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

COE Task Order 4: LLNL Final Report on Automated Target Recognition

Philip Top, Ana Paula Sales, Hyojin Kim, Stephen Azevedo, Harry Martz
Lawrence Livermore National Laboratory
Livermore, CA 94551

Work performed on the
Science & Technology Directorate of the
Department of Homeland Security
Statement of Work
HSHQPM-10-X-00045

Jan 23, 2015
LLNL-TR-665543

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

COE Task Order 4: LLNL Final Report on Automated Target Recognition

Philip Top, Ana Paula Sales, Hyojin Kim, Stephen Azevedo, Harry Martz
Lawrence Livermore National Laboratory
Livermore, CA 94551

Executive Summary

Lawrence Livermore National Laboratory (LLNL) was tasked by the Department of Homeland Security (DHS) Science and Technology Directorate (S&T) with developing an automated target recognition algorithm for finding objects of interest in checked baggage at airports. The project used x-ray computed tomography (CT) data collected on a medical scanner of various simulated checked bags. Target objects of interest include clay, rubber, and saline in various concentrations. The final results are 93.9% probability of detection (P_D) with 11.9% probability of false alarm (P_{FA}) on all targets and 100% detection for pseudo target sheets. This result is achieved by using a three-stage algorithm made up of segmentation, post-processing, and feature extraction and object classification. The segmentation proved to be the most important stage and it involves merging two distinct segmentation results followed by a set of algorithms for cleaning up the merged segmentations. The first segmentation algorithm operates on planar 10x10 voxel slabs (two- dimensional segments) in the three spatial planes. The segmenter calculates a probability that each slab in the image is a part of a target, determines if the probability was above a threshold, and then merges all connected slabs together. The second segmentation algorithm is an ensemble segmenter, which generates a set of possible segmentations through random permutations and computes an average behavior. The distinct properties of these segmenters results in a merged segmentation with superior results than either segmenter alone. The merged segmentation is "cleaned up" using post processing techniques to achieve the final labeled images. The final classification stage made use of a random forest classifier algorithm to discriminate targets from non-targets based on a set of features of the segmented objects partially from the voxel slabs and partially from the individual voxels. Features include standard statistics such as mean, median, and standard deviations of the density, features capturing some aspects of texture, and some features about the surrounding bag structure. In addition to the detection and false alarm results above, we show how the algorithms can be trained to "overfit" to the limited data set provided and achieve near perfect results (P_D=100% and P_{FA}=0%). This over-training is not recommended for long-term operation, but instead is meant to demonstrate how test design is an important factor in aviation security. Possible improvements to the algorithm and the testing methodology (including blind testing) are also presented.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Definitions

ALERT	Awareness and Localization of Explosives-Related Threats
ATR	Automatic Target recognition
AUC	Area under the curve
Bag	A package containing targets non-targets and pseudo targets
COE	Center of Excellence
CT	Computed Tomography
DCT	Discrete Cosine Transform
DHS	Department of Homeland Security
LLNL	Lawrence Livermore National Laboratory
Non-target	Something the ATR should not detect
PD	Probability of detection
PFA	Probability of false alarm
Precision	The fraction of a labeled detection which matches a real target
Pseudo-target	A target material with sub-minimum mass, sub-minimum thickness or a another material with density less than water
PT	Pseudo-target
Recall	The fraction of a real target which is captured by a labeled detection
ROC	Receiver Operating Characteristic curve
Segment	A labeled collection of voxels
Object	A segment which is intended to correspond to a physical target or non-target
S&T	Science and Technology Directorate of DHS
Target	Something the ATR must detect
TO4	Task Order 4
Voxel	A single volumetric pixel of a CT image

COE Task Order 4: LLNL Final Report on Automated Target Recognition

1 Introduction

The Department of Homeland Security (DHS) Science and Technology Directorate (S&T) has been supporting Centers of Excellence (COE) to encourage third-party participation in support of the DHS mission. The COE known as "Awareness and Localization of Explosives-Related Threats" or ALERT centered at Northeastern University is tasked with improving effective characterization, detection, mitigation and response to explosives-related threats facing the country and the world. A Task Order for ALERT (the fourth one, or TO4) was to develop automated target recognition (ATR) algorithms for finding target objects of interest in checked baggage at airports using computed tomography (CT) scanners.

Five academic and laboratory-based teams worked independently to develop these ATR algorithms. The Lawrence Livermore National Laboratory (LLNL) was one of the five participants and has developed an automated target recognition system that is described in this report. As a way to test the performance of each of the systems, ALERT acquired x-ray CT images of known targets in luggage bins that were used by all teams. The experimental data consisted of 3D single-energy x-ray CT volumes collected on an Imatron medical CT scanner. There were 93 different targets many of which were scanned in multiple bags, configured in luggage bins simulating bags[1] that were scanned and resulted in 188 volumetric CT datasets. The bags consisted of 421 target objects, 75 psuedo-targets and 1371 not target objects. An ALERT team acquired the data and identified the known "ground truth" by hand-labeling the objects in each image. ALERT also provided to the teams a common scoring program that accumulated the ATR output into a tabular format for cross-comparison. More details of the experimental methodology are available in (ALERT, 2014) .

For this project, the target objects of interest included clay, rubber, and saline in various concentrations, and a powder. The objects were placed in the bags in various shapes and configurations including blocks, bags, sheets, and other amorphous forms. The objective was to start with reconstructed voxel image data and create a set of labeled segments of the images where each label corresponds to a detection of an object of interest (target or non-target). According to the scoring criteria, a detection is counted if the label segment has a precision greater than 50% and a recall greater than 50% for bulk objects and a precision of 20% and recall of 20% for sheet objects. (Sheets, being more difficult to isolate for CT, were treated differently.) For pseudo-target sheets the precision and recall requirements were 10% for each. Precision is defined as the percentage of the labeled segment that overlaps with the labeled ground truth. Recall is the percentage of the labeled ground truth object of interest that is correctly labeled as an object of interest. The bags/bins were also packed with other objects such as water, electronics, other types of rubber, clothes and assorted other common objects. A full description of the test plan

[1] Since these bins were simulating luggage, we will refer to them as "bags" throughout.

is available in (ALERT, 2014). The available data consisted of 188 bags with various targets and labeled images with the ground truth. The ground truth was determined with a semi-manual process. All 3D reconstructed volumes and ground-truth labeled volumes were made available.

The final results with our system achieved 93.9% probability of detection with 11.9% false alarm with 100% detection for pseudo-target (PT) sheets. The full results are shown in Table 1.

Table 1. Final Results from Automatic object recognition at LLNL.

				No special rules (except for PT sheets)	
Object Type	Object Subtype	Level of Difficulty	Num Objects	Num Detected	PD [%]
Target	All	All	407	381	**93.6**
Target	Clay	All	111	107	96.4
Target	Rubber	All	158	150	94.9
Target	Saline	All	138	124	89.9
Target	Bulk	All	270	251	93
Target	Sheet	All	137	130	94.9
Target	All	Low	77	75	97.4
Target	Clay	Low	29	29	100
Target	Rubber	Low	22	22	100
Target	Saline	Low	26	24	92.3
Target	Bulk	Low	56	54	96.4
Target	Sheet	Low	21	21	100
Target	All	High	317	294	92.7
Target	Clay	High	82	78	95.1
Target	Rubber	High	125	118	94.4
Target	Saline	High	110	98	89.1
Target	Bulk	High	201	185	92
Target	Sheet	High	116	109	94
Pseudo-target	Sheet	High	10	10	**100**
			Num Non-targets	Num FAs	PFA [%]
			1371	163	**11.9**
				Num Scans with FAs	Avg Num FAs
				110	1.57

2 Algorithm Architecture and Philosophy

A general pipeline of the detection and classification algorithm is shown in Figure 1. Overall the system consists of three stages. The first stage consists of a pair of segmenters designed with different properties and a merging algorithm. The second stage consists of a collection of algorithms for cleaning up various issues with merged segmentations and specific algorithms for detecting and extracting sheets. The third and final stage contains a feature extracter for the labeled objects and a classifier to discriminate targets from non-targets.

A set of design principles guided the architecture. These included compatibility with new targets, and or different datasets; and separability or the ability to isolate and assess each individual algorithm components and replace it independent of the other components. Each stage of the process pipeline applies additional information extracted from the training data set to improve the final results and was examined and tested in isolation and in its relation to the entire process. Separability was maintained by defining an interface for each step, typically consisting of output files, and a testing framework that evaluated the effect of each individual algorithm on the results. Each stage of the algorithm could read in a file and output a file that could be later examined and tested or work in conjunction with the rest of the pipeline.

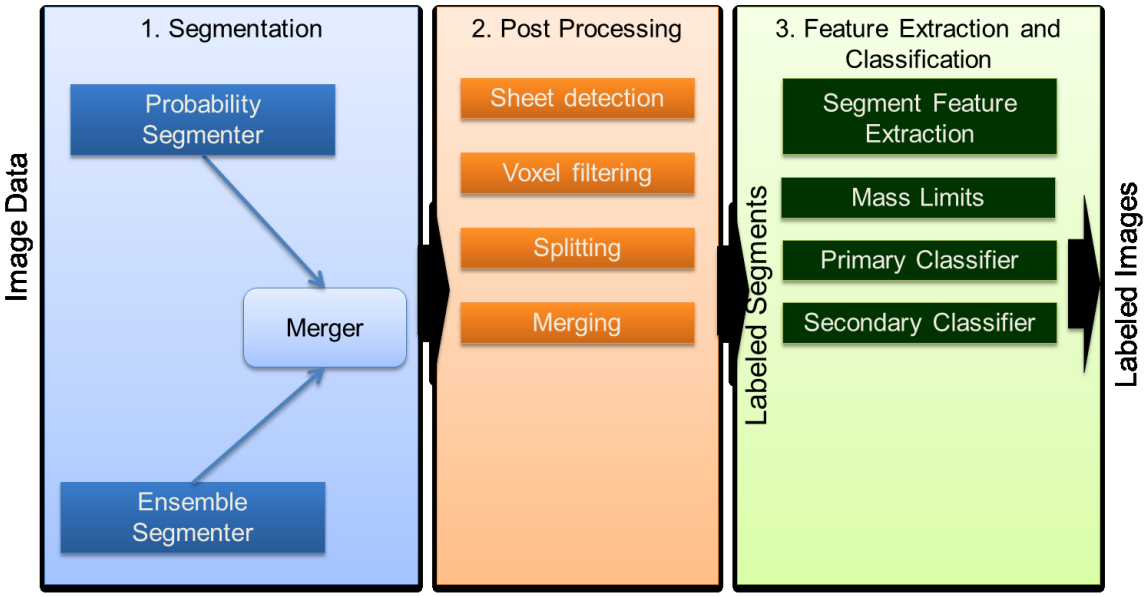


Figure 1. The ATR pipeline consisted of three stages.

3 Stage 1: Segmentation

The most complex part of the algorithm is the use of two independent segmenters. Early on in the development process it was determined that in order to meet the precision and recall specifications for this test the segmentation had to be as accurate as possible as

future steps would rely on having accurate segments from which to extract features. It was also determined that the segmenter had to include some degree of classification in order to minimize false alarms. Based on the initial explorations, two distinct approaches were undertaken for the segmentation. The first, denoted herein as "probability segmenter", described in Section 3.1, was developed specifically for this project and the second, the "ensemble segmenter", was based on ongoing research into segmentation at LLNL that was developed prior to this project and tuned for the purposes of this project and is described in Section 3.2.

3.1 Probability Segmenter

The basic premise of the probability segmenter consists of labeling the voxels that have a high probability of being part of a target and removing all other voxels from further consideration. It operates in four steps:

1. Separate all the voxels into 10x10x1 voxel slabs and compute a set of features for each of them (further details are provided in Section 3.1.1);
2. Compute the probability that each voxel slab is a member of a specific target type based on the set of features (further details are provided in Section 3.1.2);
3. Apply a threshold on the probabilities in order to map them into binary values that indicate whether or not a voxel slab is a member of a target.
4. Connect all voxel slabs above a threshold into individual labeled segments. Voxel slabs with probabilities below the threshold are discarded.

The COE task established that in order for a labeled segment to be considered a detection, the segment had to capture at least 50% of the labeled ground truth voxels for bulk objects and 20% for sheet objects. Therefore, the aim of the probability segmenter is to identify at least 50% of the voxels from all bulk objects of interest as potential targets, and at least 20% for sheet targets. To improve operation of the segmenter, an additional internal goal for this algorithm was adopted which was to capture at least 70% of the voxels from 70% of the objects of interest. This second goal ensured that the more easily identifiable objects are segmented cleanly.

3.1.1 Voxel Feature Extraction

The image voxels were separated into 10x10x1 voxel slabs aligned with each of the three planes of the 3D image with 50% overlap. An illustration of these slabs in all three planes and the outlines of the different slabs are shown in Figure 2. Each slab consisted of 100 voxels, representing a tradeoff between statistical information and target resolution. Aligning along the three dimensions coupled with the small size of the slab allows detection of sheets in any alignment. In the case of the sheets, this approach is capable of capturing enough data of the thinnest target sheet in this test even at 45 degree angle with respect to the image alignment if the slabs are overlapped. This is based on the given dimensions of the minimum sheet size and the given voxel dimensions. The voxel dimensions are 1.5-by-1.5-by-1.5 mm. The minimum sheet width was 1/4 inch or 6 mm. At a 45 degree angle the sheet would occupy five voxels along one dimension, which covers half the voxels in some slab, allowing for statistics such as the median to be

calculated based on these voxels. Some pseudo-targets had dimensions smaller than the listed dimensions in mass or thickness. The minimum target mass 250g.

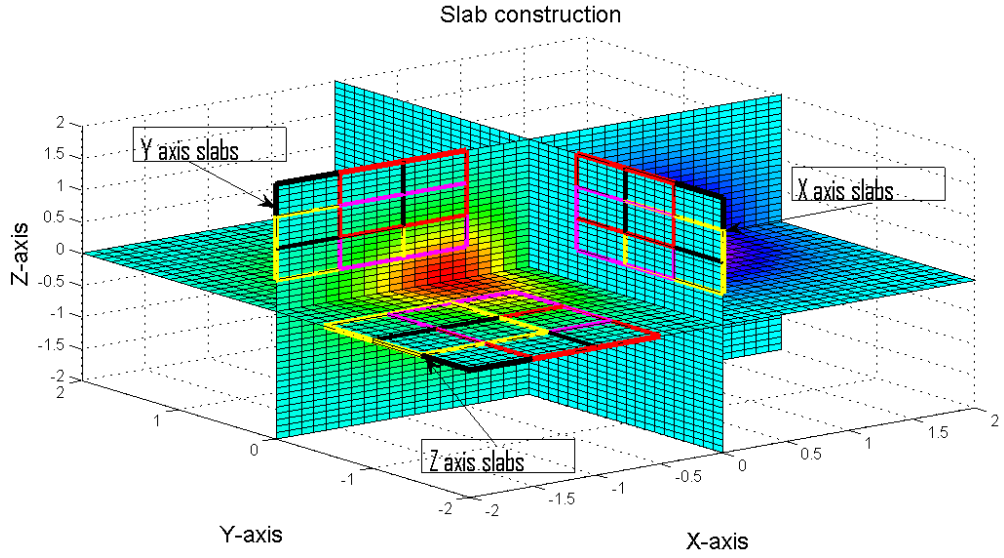


Figure 2. Voxel Slab Construction was performed in all three planes of the reconstructed CT volume.

For each of the slabs a set of 13 features was computed. The mean, median, mode and trimmean capture the density of the slab. The trimmean is the mean of the middle 90% of the data. The standard deviation, range, and 90% range capture the total variation in the slab. Six features based on the discrete cosine transform (DCT) of the data capture some aspects of the texture of the slab. Similarly to the Fourier transform, it uses sinusoidal functions as a basis for decomposing data. The Fourier transform uses complex exponentials, whereas the DCT uses only cosine functions and has the advantage of only producing real valued results. Taking the DCT of a 10x10 image produces a 10x10 result. For future discussion the element at position [i, k] will be referenced as dctik, so the element at position [2, 2] will be dct22. The dct11, or the result at position [1, 1], is equivalent to the mean value. The values along the diagonal capture the non-directional spatial variation in the data, which is primarily what we are interested in. Six features were extracted from the DCT: dct22+dct33, dct44+dct55, dct66+dct77, dct88+dct99, dct1010, and the sum(DCT)-trace(DCT). The trace of a matrix is the sum of the diagonal elements. The last feature captures all off-diagonal elements of the DCT. The diagonal elements represent rotationally invariant textural features. A detailed study was undertaken to identify the best classifiers and best features to use in discriminating between the different target types. The full details are available in Appendix A.

3.1.2 Voxel Classifier

In typical bag images, the majority of voxels are associated with non-target objects. The goal of the voxel classifier is to filter out the non-target voxels. The motivation for this step is two-fold. First, removing most of the non-relevant voxels should improve the accuracy of the segmenter, by reducing the number of decisions it needs to make. Second, by reducing the data to a small fraction of its original size, the complexity of the following steps in the segmenter, as well as in the subsequent steps of the ATR pipeline,

are also reduced. Hence, an effective voxel classifier should make the entire ATR pipeline faster and more accurate.

The voxel classifier takes as input a vector of features for each voxel slab and produces as output the probability that the voxel slab belongs to a target object. By applying a threshold to the vector of probabilities for all voxel slabs, we obtain a binary value indicating whether or not each voxel slab belongs to a target object. The voxels that are classified as non-target are removed from the image and from any further processing. The remaining sparse image is then sent forward to the next step in the probability segmenter, the connected component algorithm. The goal of the voxel classifier is to remove as many non-target voxels as possible, while retaining the vast majority of the target voxels.

The voxel classifier consists of four distinct classifiers, one for each target subtype: clay, powder, rubber, and saline. Each of these classifiers takes as an input a vector of features of a voxel, and produces a binary output indicating whether the voxel is a target of the corresponding subtype or not. Voxels that are classified as a target by at least one of the subtype classifiers are kept in the image, and the voxels that are considered non-target by all classifiers are removed from the image. A diagram of the voxel classifier is shown in Figure 3.

The voxel classifier training is a supervised step, meaning that it learned the features associated with targets versus non-targets from labeled training data. Training data consisted of a small fraction(2%) of the voxel slabs associated with labeled target data. Inaccuracies in the ground truth can lead to errors in the training set but in this case the number of samples is large that the effect of the mislabeled voxel slabs is minimal. An exception to this is noted later for the case of some pseudo-target sheets. The training data itself consists of a set of training examples, in this case, a set of voxel slabs, each of which is represented by a label (target/non-target) and a vector of features. Once the voxel classifier is trained, it can then be used to classify new data. Training involves generating a set of decision trees using random choices for splits and only a small subsample of the training data each of which is setup to optimally classify the subset of data it is given. We explored a variety of supervised machine learning algorithms (see Appendix A), and obtained the best results using random forests (Breiman, 2001), which is the algorithm that we included in our ATR pipeline as the voxel classifier.

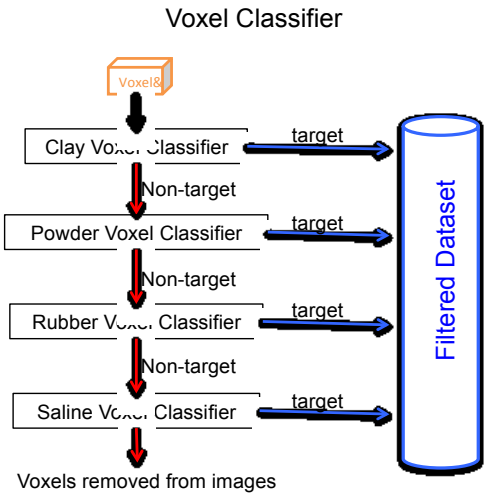


Figure 3: The voxel classifier is composed of four random forests, one for each target subtype: clay, powder, saline, and rubber. A voxel is classified as target if it is classified as positive by any one of the four random forests. Those voxels are kept in the images that are then subsequently sent to the segmentation step of the ATD pipeline. Voxels classified as non-target by the voxel classifier are removed from the images.

3.1.2.1 Random Forests

A random forest is a classification algorithm that consists of an ensemble of decision trees, where each tree is built with an element of randomness as described below. Each decision tree in the forest is a classifier that takes as input a feature vector and produces as output a binary classification, in this case, target or non-target. The random forest classifies a voxel slab by submitting its feature vector to each one of the trees and subsequently aggregating the outputs of all the trees into a final binary classification.

Decision trees operate by recursively partitioning the feature space of the data into exhaustive and mutually exclusive partitions. Each one of the partitions is based on a single feature and is associated with a label, in our case, target or non-target. Partitions are called "nodes" in the tree, and terminal nodes are called "leaves." Given a training data set, a decision tree is built by recursively finding a combination of feature and threshold that best splits the data, where best is defined by how pure the derived partitions are in terms of labels. This process is greedy, that is, at each step the combination of feature and threshold that generates the best partition is chosen. This splitting is repeated recursively for each child partition until the partitions are pure (all training observations have the same label) or until a stopping condition is reached, such as a minimum leaf size Classification of a new observation is then obtained by interrogating its feature vector at each split until a leaf is reached. The new observation is then labeled based on the label associated with that leaf. Figure 4 shows a diagram of a simplified decision tree.

In random forests, each decision tree differs from all others to different degrees. This diversity is obtained by introducing randomness to the growing of the trees, as follows. Given a training set with N observations, each tree is grown using a training set consisting of N observations sampled at random and with replacement from the original

training set. Given a feature vector of length M, at each split of each tree, a number $m << M$ of features is randomly selected out of the M features and the best split is chosen based on only the m features. Together, these two steps allow for diversity within the random forests, such that the ensemble is both more robust and effective than the individual parts. Inherently a single tree is highly over-trained and specific to a single training set, however, the random sampling of the training set and aggregation of multiple trees results in a much more robust classifier.

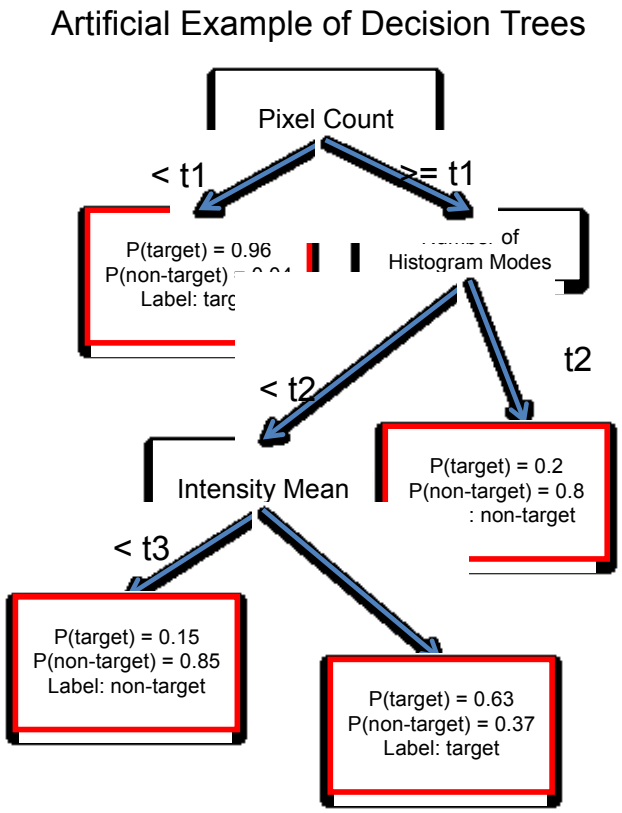


Figure 4: Decision Tree Example. Internal nodes are represented as boxes with black borders, and leaves as boxes with red borders. The labels within the internal nodes indicate the feature used at that node to split the data. Decision trees provide a partitioning of the feature space of the data into disjoint sets. Each partitioning is associated with a probability vector of the possible outcome classes. Classification of a new observation is obtained by mapping the features of the new observation to the partitioning of the data, until the new observation is associated with leaf. The class assigned to the observation is that associated with its leaf.

In addition to producing the best results, we chose to use random forests as the voxel classifier for a number of other reasons. Random forests are fast, they scale well both with number of observations as well as with the number of features. They are robust to anomalies. They account for interactions among features and non-linear relationships. Feature selection is obtained automatically, as part of the growing of the trees, and measures of relevance of each feature can be easily computed. Overall, random forests are very effective classifiers and we use them in our pipeline as the basis for a segmenter.

Another benefit of random forests is that there are relatively few parameters to tune compared to other classification algorithms. There are three primary parameters that

affect the performance of random forests: the number of trees, the minimum size of terminal nodes, and the number *m* of features tested at each split. For each of these parameters there is a trade-off. Increasing the number of trees, up to a certain limit, increases the accuracy of the forest, but also increases the computational burden, both in terms of memory and computing power. Decreasing the minimum size of leaves allows for stronger individual trees at the detriment of computational performance. Finally, increasing *m* increases the strength of the trees (i.e., makes the individual trees better classifiers) but it also increases the correlation of the trees, making the ensemble of trees as a whole weaker. We have performed sensitivity analyses to understand how these parameters affect the performance of our random forests and to tune the forests used as our voxel classifiers. Some of the results of this exploration are described in Appendix A.

To summarize, the voxel classifier is composed of four random forests, one for each target subtype (powder, saline, rubber, and clay). Each random forest is composed of 500 trees, where the minimum size of the leaves is one. The number of variables tested at each split, *m*, is three. The details and rational behind each of these parameter choices is discussed in Appendix A. The voxel classifier labels as targets voxels that are classified as positive by at least one of the random forests. Voxels classified as target by the voxel classifier are kept, and those classified as non-targets are removed from the image. The sparse images containing only voxels classified as targets are then sent forward to the next step in the ATR pipeline, image segmentation.

3.1.2.2 Voxel Classifier Results

The results presented here were obtained using a voxel classifier, as described in previous section: four random forests (one for each subtype), each of which containing 500 trees, and having minimum leaf size of one and three features tested at each split. The complete set of features provided as input to the random forests consisted of the following summary statistics for each voxel slab: mean, median, standard deviation, range, dct22+33, dct44+55, dct66+77, dct88+99, dct1010, and dctA-trace. As described in Section 3.1.1, the features starting with prefix "dct" are elements of the inverse cosine transform. Overall, for all four subtypes classifiers, the mean and median of the voxel intensities were the most informative features (see Figure 5), followed by standard deviation and range. In the case of rubber the range is particularly useful likely due to the relative consistency of the density in the rubber objects Figure 10 shows the distribution of the range feature for the various object types.

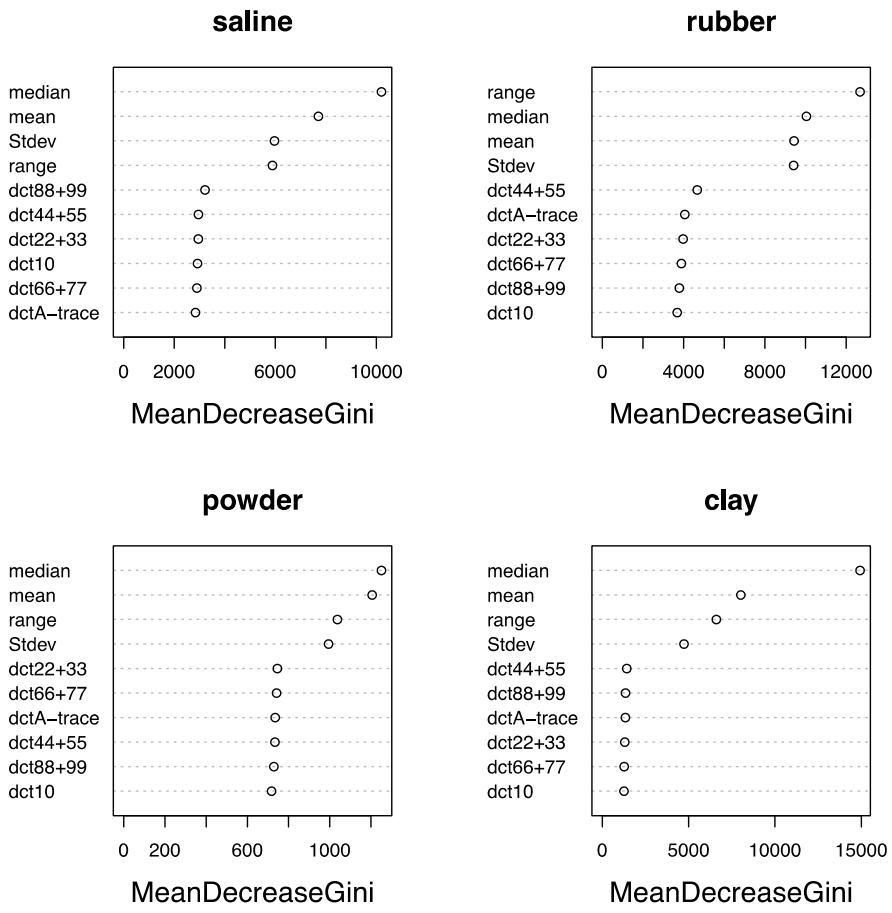


Figure 5: Relative importance of features for each of the four random forests (saline, rubber, powder, and clay) of the voxel classifier. The mean decreased Gini coefficient is a measure of how each feature contributes to the homogeneity of the nodes and leaves of the trees in the forest. The higher the mean decreased Gini coefficient of a feature, the more important it is for the random forest.

Table 2: Percentiles of number of voxel slabs of objects of each class that were correctly classified by the corresponding classifier.

	saline	rubber	powder	clay
0%	0.78	0.69	0.76	0.01
10%	0.82	0.72	0.82	0.65
25%	0.86	0.85	0.87	0.88
50%	0.92	0.91	0.89	0.93
75%	0.95	0.94	0.92	0.96
100%	0.98	0.98	0.98	0.99

In order to minimize over-training on the data, we perform three-fold cross-validation. In other words, the training data is randomly partitioned into three complementary sets, such that one of the sets is used to train the algorithm, which is then applied to evaluate the

data in the remaining two sets. Using this scheme, voxel slab features are only evaluated by random forests that have not seen that observation while being trained.

All four subtype classifiers produce results with relatively high accuracy, with all of them having an area under the performance curve(AUC) at or above 0.9. AUC is the total area under a plot of the PD vs PFA created by varying the threshold at which a voxel slab is classified as target vs non-target and is a measure of the classifier performance. Overall, the best results were obtained for the clay classifiers (AUC ~ 0.97), and worst results for powder classifiers (AUC ~ 0.899), as shown in Figure 6. Table 2 shows the quantiles of the percentage of voxels of objects of a given class (saline, rubber, powder, and clay) that were correctly classified by the corresponding classifier. For saline, rubber, and powder, all objects had at least 69% of voxels correctly classified. The median of the fraction of voxels of objects correctly classified by the corresponding classifier was 0.92, 0.91, 0.89, and 0.93, for saline, rubber, powder, and clay, respectively.

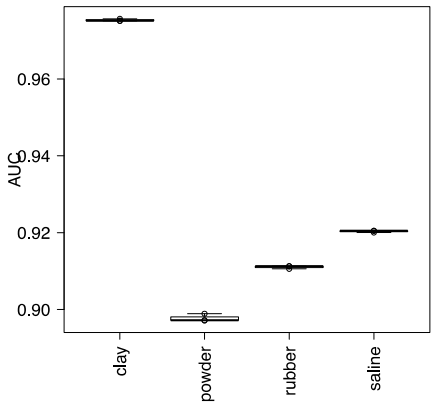


Figure 6 AUCs obtained by classifiers for the four object subtypes: saline, rubber, powder, and clay. Each box represents three AUCs, one for each of the three cross validation sets.

On a per-object basis, all target objects, except for one (object id = 7011), had at least 84% of its voxels included in the filtered set. Object 7011, a bulk object consisting of two blocks of clay merged, had 56% of its voxels included in the filtered dataset. The percentage of voxels of each object from the cleaned set retained in the filtered set is show in Figure 7. Objects 8024 and 8026 which were pseudo-target sheets were also characterized by lower detection rates likely due to the lower accuracy of the ground truth in those objects.

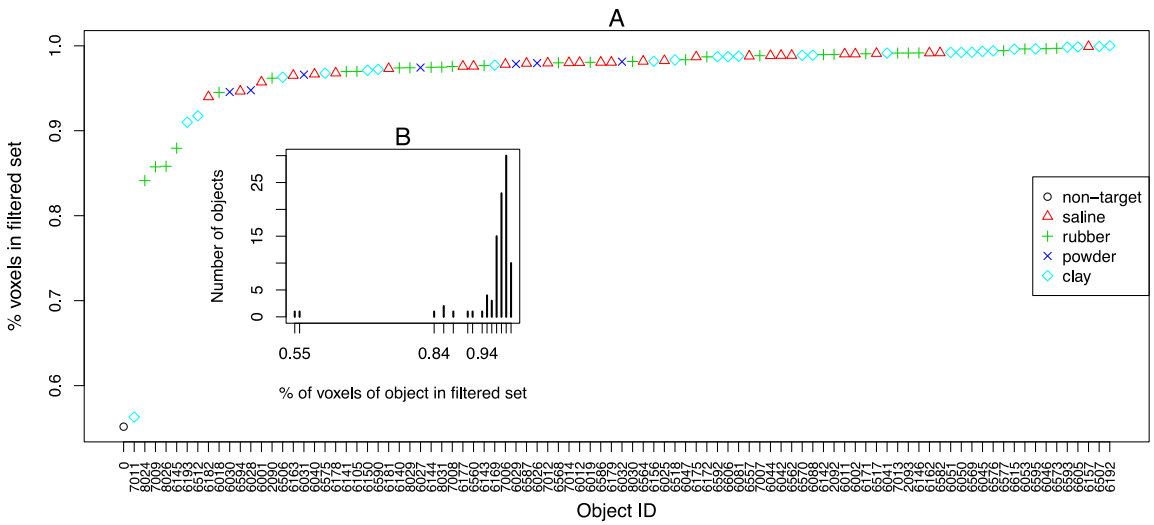


Figure 7: (A) Percentage of voxels of each object from the original dataset included in dataset filtered by the voxel classifier colored by object type. (B) Histogram of number of objects distribution as a function of the percentage of voxels included in filtered set.

3.1.2.3 Voxel Classifier Implementation

Using the information from the classification analysis we trained four classifiers using a subset of the calculated features. All four classifiers used the median, standard deviation, and range, and one of the DCT features. The rubber classifier used dct88+dct99, the saline and clay classifier used dct44+dct55 and the powder classifier which was later disabled since we were not required to detect powders, used dct22+dct33. Probability density functions for the various properties are shown in Figures 8-13. These figures illustrate that while the median is clearly the dominant feature for discrimination, the other features do provide some additional discrimination capability. Improved discrimination might be possible with ratios of DCT elements but this potential was not fully explored in the course of this project.

In order to better detect some pseudo target sheets and a few other anomalous objects it was necessary to select some specific bands in the rubber classifier and clay classifier to ensure sufficient detection of those objects. All voxel slabs with median intensity values of [1060 to 1150] for rubber and [2200 to 2300] for clay were labeled as high probability detections. The likely cause is due to some mislabeling in the pseudo target sheets along with a different type of rubber, and some deviations in the specific objects in question from the normal "clay-like" behavior.

Final results from the probability segmenter indicated that the high threshold captured at least 60% of the voxel slabs from 69% of the targets with a false alarm rate of 9%. A majority of these false alarm voxels are contained in a few objects or were removed, as they were isolated from other potential targets. The low threshold met the goal of detecting all targets at the required thresholds with a false alarm rate of 16% of the voxels in a bag. After the connected component segmentation numerous objects were merged together which reduced the PD significantly. A more sophisticated method of segmentation could have been used but it was not necessary when this method is used combination with the rest of the system.

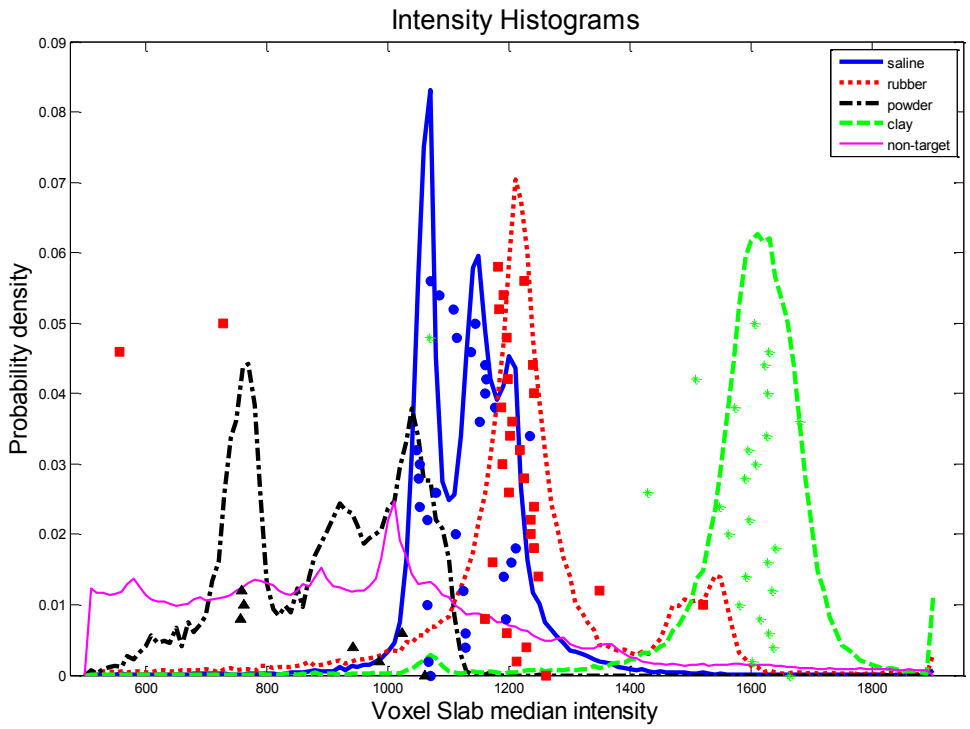


Figure 8 Voxel Slab Median Value Distributions

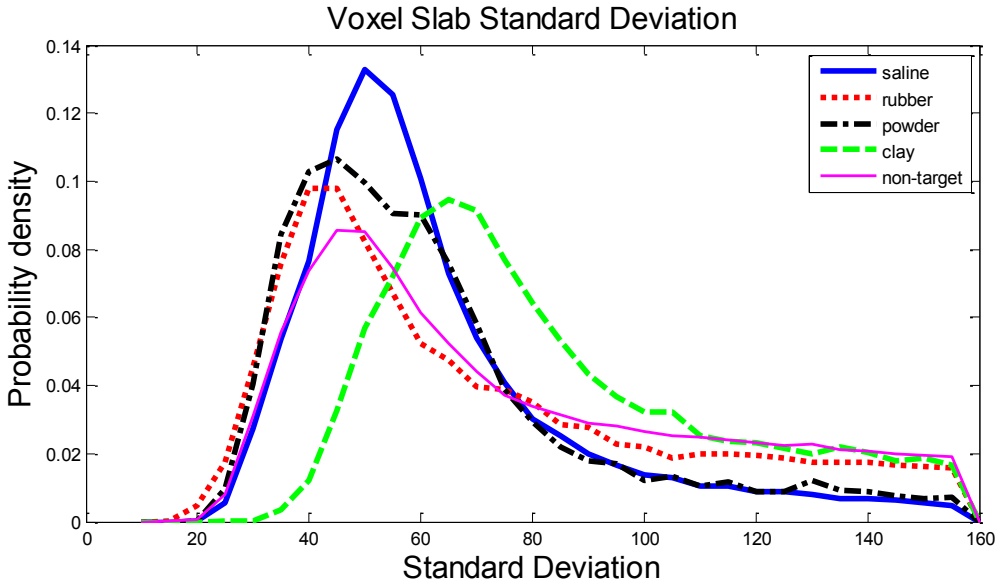


Figure 9 Voxel Slab Standard Deviation Distributions

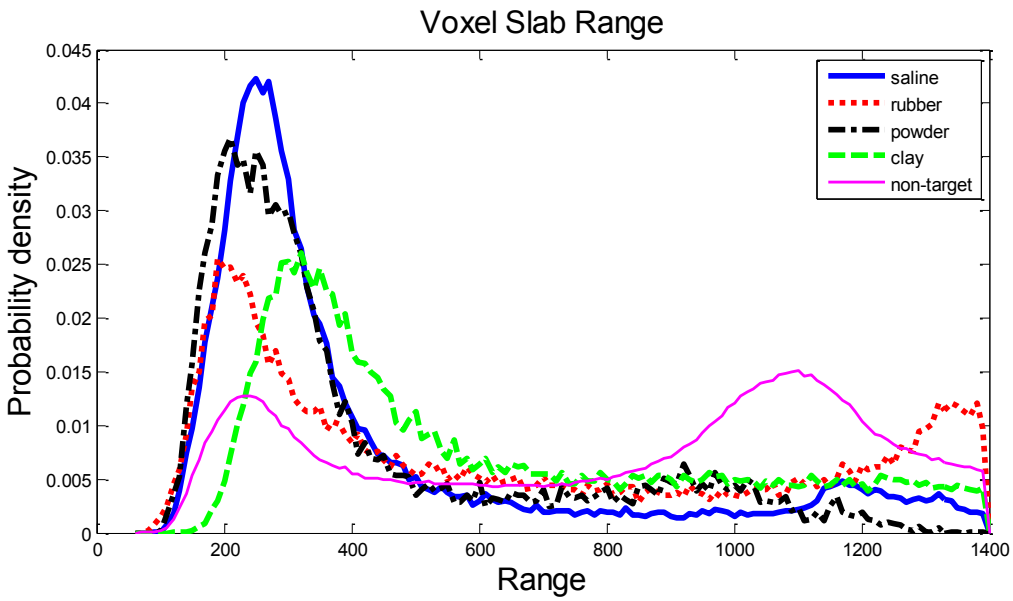


Figure 10 Voxel Slab Range Distributions

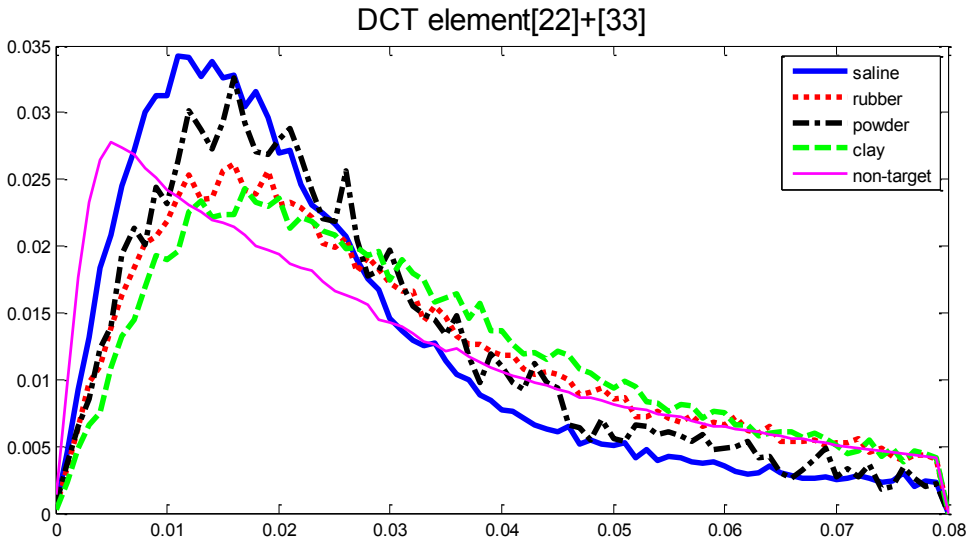


Figure 11 Voxel Slab dct22+dct33 distributions

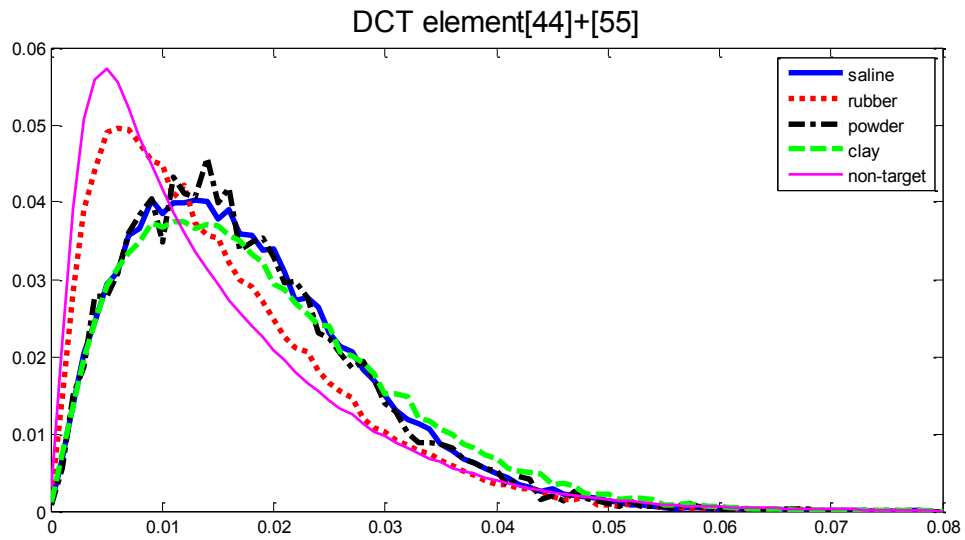


Figure 12 Voxel Slab dct44+dct55 distributions

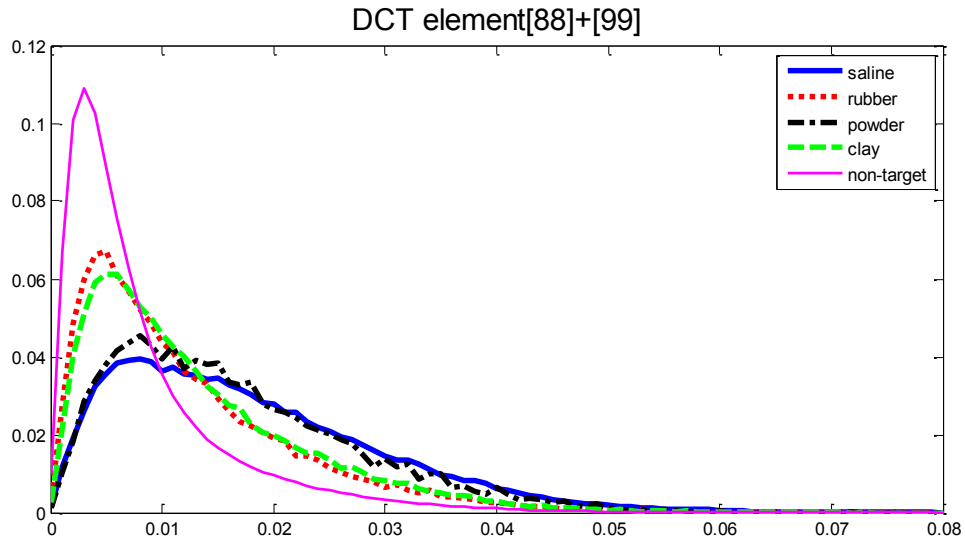


Figure 13 dct88+dct99 distributions

Each voxel slab was fed into each of the different classifiers and reported as a probability from 0 to 1 that the voxel slab contained target voxels. As part of the training, two thresholds were chosen for each target type. The higher threshold captured 80% of the voxels from 90% of the targets and the lower threshold captured 50% of all the targets based a small training set of data.

3.1.3 Connected Components Algorithm

The voxels identified as having probabilities above the high and low thresholds were merged using a connected components algorithm for each of the target types. The connected component algorithm identified all voxels as having neighbors in any of the three dimensions as being connected and part of the same object. Isolated voxel slabs typical of noisy pixels were removed from the labeled set. Single planes of voxel slabs were also removed from the connected object as this type of structure was typical of

some forms of x-ray reconstruction artifacts namely metal streaking artifacts. The removal of these artifacts is an area that could benefit from additional analysis. For each target type areas with high probability were segmented and labeled before repeating the process with the low threshold areas. Segments that triggered thresholds for multiple target types were merged or split as appropriate. Segments formed from the higher threshold were kept separate while low threshold segments were merged with segments from other target types. This process produced a set of labeled segments for each image, resulting in a PD of 61% and a PFA of 91% of the final targets. The primary reason for the low positive detection and high false alarm rates was due to the fact that many targets were merged together by the simple segmentation process. Due to the constraints of the official test metrics (specifically that segments must meet the 50% recall and precision rates), merged targets can simultaneously reduce the positive detection and false alarm rates. Although the performance of the probability segmenter alone at this stage was underwhelming, when combined with the ensemble segmenter (see Section 3.2), it leads to improvements in both positive detection and false alarm rates, as discussed in Section 4. The purpose of this segmenter in the merged pipeline is to complement the ensemble segmenter. If it was operated alone additional complexity in the segmentation process and post-processing could improve the results significantly, but this was unnecessary for this test when merged.

3.2 Ensemble Segmenter

*

Note that this segmentation work was performed under the support of LLNL internal R&D funding (Laboratory Directed Research and Development Program) for "Coupled Segmentation of Industrial CT Images." The developed segmentation approach was applied to the problem of airport security and more specifically to TO4 data. We submitted a paper about the algorithm and experiments using TO4, which is currently under review and this section was excerpted from the paper (Kim, Thiagarajan, & Bremer, (Under Review) A Randomized Ensemble Approach to Industrial CT Segmentation, 2015):

We developed a highly flexible segmentation approach that uses an ensemble of hierarchical segmentations and exploits high-level semantic information to effectively find objects. This approach builds multiple hierarchical segmentations to explore as many potential ways of segmenting as possible, by randomizing the merging order of segment region pairs. Among potential segments in the ensemble, the most likely candidate regions are filtered, by incorporating high-level features defining objects. Finally, all localized candidates are combined into a consensus segmentation to produce the final segments.

This segmentation ensemble approach has several advantages: First, the multiple randomized segmentations, some of which are expected to be accurate or sufficiently close, compensate for a large variety of noise and artifacts; second, the global search for likely objects allows segments at multiple levels of the hierarchy; third, using a simple

reference-based scheme, we compare segments to the training data, and robustly identify a set of candidate segments likely to describe objects of interest; Lastly, sequentially localizing all candidates for a particular object/region, the per-object consensus segmentation performs graph-cut segmentation to obtain a final object region, without the need to directly estimate the number of objects in the candidates.

3.2.1 Algorithm Overview

The algorithm begins with supervoxel-based over-segmentation to partition the entire volume into small-sized atomic regions, referred to as supervoxels. Then we construct an ensemble of bottom-up hierarchical segmentations from the initial set of supervoxels. Each hierarchy incrementally merges regions from the previous level. The edge affinity is measured as the similarity between their intensity histograms. Suppose $w^l_{i,j}$ is the edge affinity between two regions r^l_i and r^l_j in hierarchy level l. We compute $w^l_{i,j}$ as $w^l_{i,j} = exp(-\sigma \chi^2(H(r^l_i), H(r^l_j)))$ where $H(r^l_i)$ is the intensity histogram of region r^l_i, and χ^2 measures the chi-square distance between two histograms, and σ is the parameter for the Gaussian radial basis function.

To generate multiple independent segmentations from the same set of supervoxels, we randomize the merging order of candidate edges in each hierarchy level. See the paper (Kim, Thiagarajan, & Bremer, Image Segmentation using Consensus from Hierarchical Segmentation Ensembles, 2014) for more details about how to construct multiple randomized hierarchies.

Among all potential segments from the ensemble, we extract candidate regions likely to be an object of interest. For each potential segment, we first compute its feature set that consists of intensity statistics, shape features, area, and volume-to-surface area ratio. We then extract a semantic descriptor from the feature using local discriminant embedding (LDE), a supervised graph-embedding approach, to compare with supervisory data to determine whether or not it is a good candidate.

The final step is to segment multiple objects, given the candidate regions extracted from the segmentation ensemble. We segment object regions sequentially in the order of their confidence measures until all available candidates are examined, in which we do not need prior knowledge of the number of segments. We first sort all the candidate regions in the decreasing order of their confidence. We then pick the first available candidate region with the highest confidence and collect other available candidates that have a high volume overlap ratio with the first region. The collected candidate regions are combined into a graph-cut segmentation to obtain a final consensus object region. These regions are excluded from the candidate regions, and we continue the per-object segmentation if there is still available candidate region. Figure 14 illustrates an overview of the proposed approach.

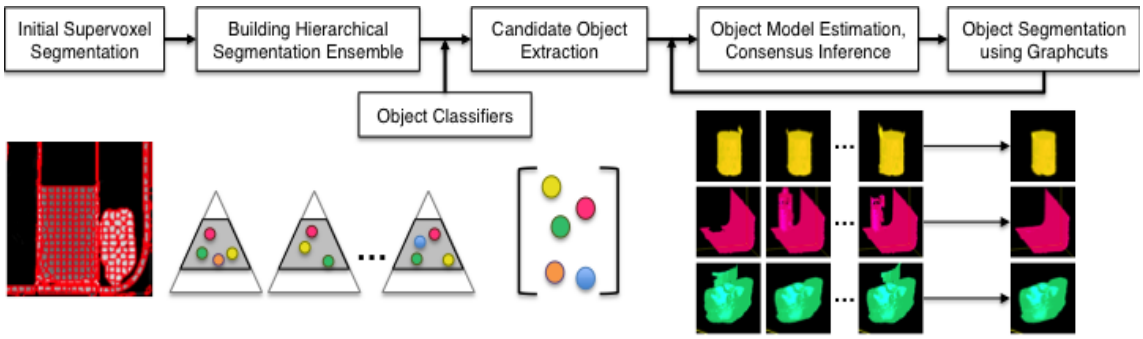
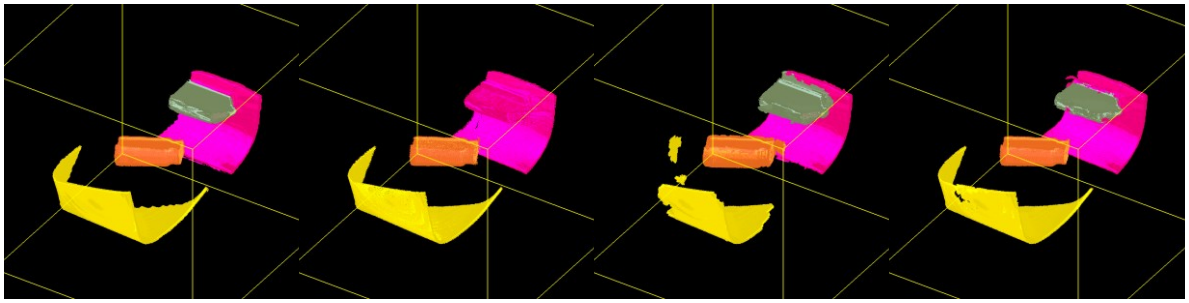


Figure 14 An overview of the ensemble segmenters

3.2.2 Experiments using TO4 Data

We applied our segmentation technique to the automatic target recognition (ATR) systems in the problem of airport security using the TO4 dataset. Here we concentrated only on the segmentation stage, aiming to provide a highly accurate segmentation of all target objects, without the target classification. For the quantitative evaluation of our segmentation, we trained our object classifier (an internal classifier within the ensemble segmenter system) using ground-truth labels of 4 different targets and pseudo targets. We extracted object features of all ground-truth regions and then extracted their LDE-based semantic descriptors. These semantic descriptors were used for extracting candidate object regions, as described previously.

In our setup, the only free parameter is the number of hierarchical segmentations, which was fixed to 20. We evaluated the segmentation performance of the proposed ensemble segmenter, in comparison to existing techniques: the popularly adopted region-growing technique and a semi-supervised graph-cuts approach, as shown in Figure 15. For the region growing, we manually tuned parameters for the TO4 dataset. For the semi-supervised graph-cuts, we simulated user's manual intervention by dilating the ground-truth label regions, and applied graph-cuts in each of those regions. Results from the scoring mechanism indicated and overall PD of 92% and a PFA of 45%.



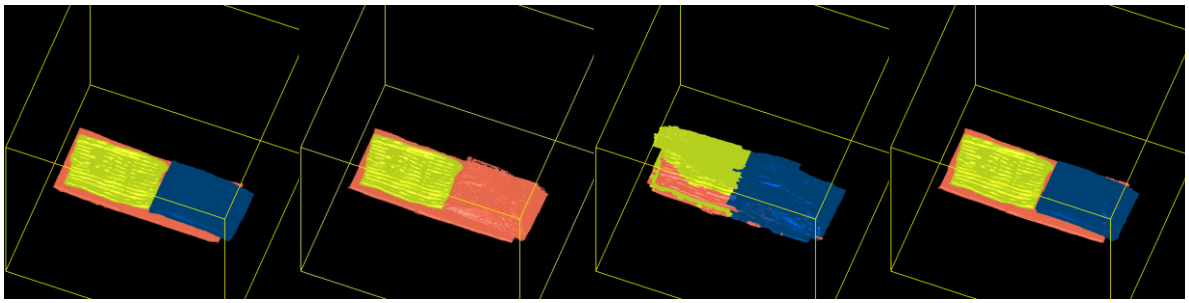


Figure 15 Segmentation results (SSN: 076, 148) of the proposed algorithm with other methods. From left to right in each row, the ground-truth labels, region growing, semi-supervised graph-cuts, and our ensemble segmenter

3.3 Segmentations Merger

The final step in the segmentation process is to merge the probability segmenter results with those from the ensemble segmenter. The process chosen takes the segmentation from the ensemble segmenter and adds new segments where the probability segmenter results contain labels not present in the ensemble segmentation result. This process resulted in a number of extra segments and small pieces but these were straightforward to remove of merge in the subsequent processing steps.

Merging segmenters in this way improves the overall results when combined with the post processing steps due to the different properties and characteristics of the different segmenters. The ensemble segmenter generally has better looking segments when graded by visual inspection but it can miss targets completely, particularly in the case of sheets. The missed targets represent a cap on the performance of the ensemble segmenter. Though the segments it does label are generally superior to the segments of the probability segmenter for the same object, the probability segmenter does not have the same theoretical performance limitations. Therefore, merging the two segmenters can raise the performance ceiling of the overall system. The downside of the probability segmenter is its tendency to merge many objects together, which is why the simple approach of basically using the ensemble segmenter results and adding any additional segments from the probability segmenter to the merged result tends to work. If it was desired to use the probability segmenter alone, additional complexity in the segmentation and post-processing steps would be needed to improve the results. Adding more complex merging algorithms into the merger step did not lead to any substantial improvement in the final result so they were not included in the final system.

4 Stage 2: Post Processing

Following the merge step was a series of post-processing algorithms designed to refine the segments. Numerous of examples of this type of processing are found in the existing literature on this subject. The patent from Telesecurity (Kwon, Lee, & Song, 2013)applies many similar techniques and further examples are found throughout the project report from TO3 (Crawford, Martz, & Pien). Sheet detection is also a topic of related patents (Eberhard & Meng-Ling, 1998). The distinguishing feature of the post-processing steps developed for this project is that they operate on voxel slab data vs. the original image data. This feature simplified the process significantly and allowed better

characterization though at the expense of a small degree of resolution in the final segment. The voxel slabs were recalculated at the beginning of the process using solely voxel data from each label. The slabs started from the lowest indexed voxel from each label and progressed in steps of 10 for each dimension. Overlapping voxels was no longer necessary as experiments with different edges or overlapping blocks showed no change in the final results calculated by the scoring algorithms

Post processing consisted of a series of algorithms to split segments followed by a set of algorithms to merge segments. The first step is a sheet detection process. The thickness of the labeled segment is determined in each of the 3 dimensions. To be declared a sheet a segment had to have a thickness in one of the dimensions between 3 and 16 voxels for at least 25% of the segment area in that dimension and cover an area of at least 50 voxel slabs, or have thickness of between 12 and 32 voxels over 25% of its surface and cover an area of at least 80 voxel slabs. Once a sheet was detected all "clumps" on the surface were removed and declared a separate segment. This detection scheme was used to identify sheets and separate sheets from bulk objects with similar properties. A clump was identified as having a thickness greater than 2 times the sheet thickness and not be attributable to a sheet turning a corner.

The next stage is a statistical splitting routine. This algorithm computed the histogram of the median in the voxel slabs, searched for distinct peaks in the histogram and if the peaks met a set a thresholds the object was split. Specifically, the peaks had to be separated by at least 75 intensity levels, and the low point between them on the histogram had to be less than 85% of the peak counts. The split was then done at the minimum of a smoothed count value. The smoothing was done using a 20 point moving average filter. A check was then done to ensure that the voxel slabs from the different sides of the split were in fact in distinct locations in the image if so the object was split.

The third stage was a called a "cube split" The general idea is to count all the voxel slabs from a segment in a 10x10x10 cube then order these counts remove a certain fraction and do a connected components analysis on the remaining cubes. All large (>100 cubes) clusters of cubes were then split into new segments. The effect is to split objects that were close together but only touching on a thin set of points or were not touching to begin with as a result of the naïve connected components in the probability segmenter or merge process. This process achieves a similar effect to dilation and erosion which is another typical technique.

Following the split process, a final merge process was conducted. The primary statistic for merging is the mode of the histogram of medians of the voxels slabs of a segment. Distinctly labeled sheets with similar modes were checked for alignment and merged if they aligned. Then objects were checked for similar modes in the histograms, if the histograms matched and they overlapped, for large objects the shape and alignment was compared and if it matched they were merged, small objects were merged with less stringent criteria. The merging criteria were based on heuristics garnered from a subset of the data and subsequently applied to the entire data set. Based on the amount of overlap and the proximity of the histograms and the matching edges of the segment they

were merged in entirety or partially. The exact threshold depends on the size of the segment and its mode.

Once the final segments were determined the voxel slabs were converted into a labeled image. The conversion process mostly consisted of mapping the voxel slabs back onto the voxel space in blocks of 10. However, all voxels with intensity values less than 475 or greater than 2300 were cut from the final labeled segment. Small objects that could not meet the minimum mass threshold established by this project were also removed. Mass per voxel was approximated by intensity.

5 Stage 3: Feature Extraction and Object Classification

The segmentation step labels segments as targets, but there are a large number of false positives. That is, a large number of the segments represent non-target objects. The goal of the "object classifier" is to filter out these false positive segments, reducing the probability of false alarm (PFA). Hence, the object classifier takes in a list of segments, and for each of those, it determines whether it should be labeled as target or non-target. In the best case scenario, the object classifier reduces the PFA of the pipeline close to zero, while maintaining intact the positive detection (PD) rate.

To complete this process a set of features was extracted from each of the segments.

1. Mean of original image voxels
2. Standard deviation of original image voxels
3. Mean of voxel slab medians
4. Median of voxel slab medians
5. Standard deviation of voxel slab medians
6. Median of voxel slab standard deviations
7. Median dct22+dct33
8. Median ((dct44+dct55)/(dct22+dct33))
9. Median voxel slab range
10. Mean Voxel slab range
11. Median dct44+dct55
12. Median dct88+dct99
13. Mode of voxel slab median
14. Mode of the original image voxels
15. Lower half width the mode – first bin with more voxels than 0.5*mode histogram count
16. Upper half width of the mode –last bin with more voxels than 0.5*mode histogram higher than the mode
17. (Optional) 2nd histogram mode peak if present
18. (Optional) 3rd histogram mode peak if present
19. (Optional) 4^{th} histogram mode peak if present
20. Total Bag mean
21. Fraction of Bag with intensity >2300
22. Mean of segment bounding box+20 voxels in all dimensions

23. Fraction of bounding box with intensity >2300
24. fraction of voxels in segment between 1060-1150

The final feature was thought that it might help with identifying pseudo target sheets. A study similar to what was done for the probability segmenter was also done with the final stage classifier

5.1 Methodology

Similarly to the voxel classifier, for each observation the object classifier takes in a feature vector and produces as an output a binary classification: target or non-target. Whereas in the voxel classifier the input feature vector consisted of features extracted from individual voxel slabs, in the object classifier the features are obtained from the combination of all voxel slabs associated with a given segment. We used a total of 24 object features, as described above.

The object classifier consists of four random forests, each specializing in one of the four target subtypes: powder, clay, rubber, and saline. A segment is deemed to represent a target object if it is classified as a target by any one of the four classifiers. We avoid overtraining by using three-fold cross-validation. For more details on the methodology, see Section 3.1.2. The training set was built by using the provided ground truth to label the segments appropriately.

5.2 Results

One way to measure the performance of the object classifier is by comparing the performance of the labeled images produced by the ATR pipeline prior to the object classifiers against the performance of the same images filtered by the object classifier. Based on the output produced by the T04 evaluation tool, the PD for clay, rubber, and saline for the former set of images were respectively 98.2, 96.8, and 94.9, and the PFA was 43.1%. The object classifier was successful in drastically reducing the PFA, from 43.1% to 11.9%, while decreasing the PD by only very small amounts (PD for clay, rubber, and saline were respectively 93.6, 96.4, and 94.9%). Figure 16 shows the relative importance of the segment features in the classifiers. These results indicate that the mass limit is important in filtering out small segments and some measure of the object density is next. The rest of the features show some value but perhaps only in a limited number of cases. A number of examples of the detection results are shown in Appendix B along with some cases where the detection failed.

5.3 Improving Results for Corner Cases

Corner cases are situations where the correct classification requires an adaptation of the classification pipeline based on some specific property of the object being detected. The results described above were obtained by "honest" approach, where overtraining was avoided to the extent practical. This is necessary in order for the results to reflect expected performance of the ATR pipeline when applied to similar but not-yet-seen data. In order to further improve results and achieve the level of accuracy required by this competition, PD approaching 100% for all target subtypes and PFA close to 0%, we

firmly believe that overfitting is a must, and that is what in fact the vast majority of vendors end up doing in order to meet the requirements to get certified. We show here how we can include additional steps, in the form of secondary classifiers that are put in place to capture the segments that were erroneously labeled by the classifier as false positives, and hence removed from the labeled images. These secondary classifiers are designed based on the specific instances misclassified by our pipeline, and hence when applied to the same set of images that were used to create them in the first place, can improve the performance of the pipeline to the levels of PD and PFA required. In fact, by modeling specific corner cases, we were able to get the PD for all low difficulty targets up to 100%, and reduce the PFA down to 1.1% using these data-derived rules. The problem is that these results are misleading, as we do not expect to see these rates of positive detections and false alarms in the future when the pipeline is applied to a new set of images.

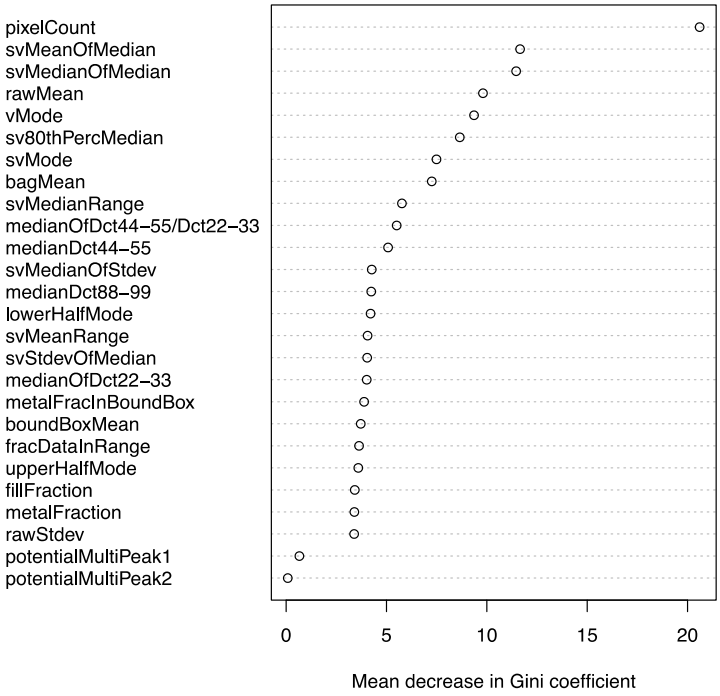


Figure 16: Relative importance of features in the construction of the object classifier. The mean decreased Gini coefficient is a measure of how each feature contributes to the homogeneity of the nodes and leaves of the trees in the forest. The higher the mean decreased Gini coefficient of a feature, the more important it is for the random forest[2].

[2] Abbreviations list. *pixelCount*: pixel count; *svMeanOfMedian*: mean of the median of contained supervoxels; *svMedianOfMedian*: median of the median of the contained supervoxels; *rawMean*: mean value of all single voxels; *vMode*: mode of the voxels; *sv80thPercMedian*: 80th percentile of the median of the contained supervoxels; *svMode*: mode of the supervoxels; *bagMean*: bag mean; *svMedianRange*: median range of the contained supervoxels; *medianOfDct44-55/Dct22-33*: median of dct44-55/dct22-33; *medianDct44-55*: median of dct 44-55 of contained supervoxels; svMedianOfStdev: median of standard deviation of contained supervoxels; *medianDct88-99*: median of dct 88-99; *lowerHalfMode*: lower half width mode (first bin with more voxels than 0.5*mode histogram count); *svMeanRange*: mean of range of contained supervoxels; *svStdevOfMedian*: standard deviation of median of contained supervoxels; *medianOfDct22-33*: median of dct 22-33; *metalFracInBoundBox*: metal fraction in bounding box;

6 Discussion

Of the missed targets, seven were due to inappropriate mergers with other objects, another 4 were due to inappropriate split targets, 1 object was split and merged with surrounding object and one object was corrupted by metal artifacts that prevented any segments from reaching the required detection threshold. The advantages of our pipeline are flexibility to new targets, data, and algorithms and ease by which new techniques and components can be integrated. Novel techniques include the use of texture based features in the probability segmenter and final stage classifier, and the use of multiple independent segmenters to improve the overall results, the use of voxel slabs for feature generation, and the use of random forest classifiers.

Both of the segmenters used contain characteristics that allow them to operate even in the presence of x-ray artifacts and thus no particular response to them was required. The probability segmenter masks over them with the use of voxel slabs and filtering of anomalous planes and the ensemble segmenter can overcome them with the help of the ensembles.

Object shapes play no role in the probability segmenter and only a minor role in the training of the ensemble segmenter. Shape was used in the extraction and detection of sheets but not elsewhere and none of the features used in the final stage classifier were based on shape. It is anticipated that similar targets in untested shapes would perform in a similar fashion to the current results.

Overtraining, except where intended, was minimized by sequestering a subset of the data, in the case of the ensemble segmenter; only using a small fraction of the data, in the case of the probability segmenter; and using three-fold cross validation for the object classifier. These methods do not entirely remove the possibility of overtraining—only limiting its influence to a certain degree. However, the nature of the project forced training to the test. The thresholds used in the probability segmenter and in some cases in the post processing methods are tied closely to the definitions of a detection and false alarm used in the experiment. In addition, many of the design decisions about where to put research effort are driven by the potential for impact on the final results.

6.1 Potential Improvements

Some initial explorations were done using complete image segmentation based on an available segmenter from Statovan (Woodhouse, June 2012). Stratovan was one of the teams working on TO3 for segmentation. The full report from this effort is available in (Crawford, Martz, & Pien). The results from the segmentation contained too many segments to be of practical use in this project. Furthermore, objects tended to be split

---

boundBoxMean: mean of the bounding box; *fracDataInRange*: fraction of the data between 1060 and 1150 (attempt to aid in pseudo sheet detection); *upperHalfMode*: upper half width mode (last bin with more voxels than 0.5*mode histogram higher than the mode); *fillFraction*: fill fraction of the bounding box; *metalFraction:* metal fraction in bounding box; *rawStdev*: standard deviation of all single voxels; *potentialMultiPeak1*: potential for single histogram peak; *potentialMultiPeak2*: potential for double histogram peaks.

into multiple segments requiring significant added complexity in the pipeline to merge them together. It is possible that with further assistance from the authors this segmenter could be better tuned to meet the requirements of this project, but for this project it was deemed impractical.

The selection of voxel slabs aligned with the 3 coordinate axes was sufficient for this project, however it is recognized that certain anomalous arrangements of targets would be problematic. There are a number of mechanisms for addressing this shortcoming. Additional planes could be formed with the 45 degree planes, this approach would likely be required if even thinner sheets were required to be detected, but it was not necessary to achieve the performance goals of this system. Sheets with thicknesses approaching the voxel dimension such as some of the pseudo target sheets are detected as long as a significant fraction of the sheet is aligned within approximately 20 degrees of one of the image axes, as they all were in this test. Though not necessary for this test further sheet specific algorithms are known and have been developed for similar applications (Eberhard & Meng-Ling, 1998). These algorithms could be applied instead of or in conjunction with the applied system.

The algorithms used in this project are all prototypes and could be refined significantly with further effort. A rigorous mathematical treatment of the probability segmenter with additional exploration of various features could lead to further reduction in false alarm rates. Likewise a rigorous treatment of the merger process and the addition of techniques previously developed in the COE task order on segmentation could also lead to further improvements. Some additional treatment of artifacts may be warranted particularly in the context of merging split object though these artifacts could also be dealt with by a better reconstruction. Most potential improvements rest in improving the segmentation results, which results in better features for the final classifier to act upon. Improvements in the overall system will be driven by improvements in segmentation as any appropriately applied modern classifier will tend to perform similarly with a given feature set. A significant potential area of improvement could rest in the refinement of the split and merge process. Cleaner segments could be achieved by further processing directly on the voxel data. Doing so was not necessary for the test criteria in place for this study.

6.2 Algorithm Discussion

As part of this project we conducted some research to determine what would be required to achieve 100% detection with 0% false alarm on the given data set as an example of how overtraining would play a role. Since there was no true blind testing which is really required to gauge true performance, the test results can be manipulated or skewed through overtraining both intentional and unintentional. While the ideal result was not achieved in practice, the achieved results come close and are shown in Table 10.

The achieved results reduced the PFA to 1.1% and increased the PD to 95.1. The distinction between the two results rests in the use of corner cases. Objects with specific properties can be treated differently than other objects throughout the detection process, as these clusters of objects become smaller and smaller at some point the final results

becomes highly overtrained to the training set and the performance results from the test bear little resemblance to the actual performance of the system. To achieve the results shown small clusters (at least two objects) were formed based on various features of the segment such as mean intensity, standard deviation, and number and location of histogram modes to distinguish merging and splitting in the post processing steps for the final segmentation. And several secondary classifiers for different object types in the object classifier stage. If this were extended to single object clusters a cursory examination indicated another six objects could be detected improving the PD somewhat more. Applying similar techniques in the segmenter could potentially detect the remaining missed detections. It is clear that this result is over-trained but the boundary between overtraining and not is not particularly clear in this context since in some cases the clusters given special treatment could be for physical reasons in other cases they could be very specific to the test set and the only way to determine that is through the use of a series of truly blind tests.

Throughout development of the pipeline presented here many design decisions were made because they did not affect the final test results based on the test criteria given. The effect is that many of the details of the final system become tied to the exact nature of the test criteria. If those criteria were different the resulting pipeline would look very different. The design process by which it was developed would be substantially similar but the final pipeline would be different. For instance if the test metric were based on actual precision and recall scores instead of a binary decision, that would have necessitated the use of more operations at the individual voxel level to gain a few extra percentage points, whereas in present form such processing had no effect on the final score and were thus left out. Many design choices such as the complexity of the segmentation merge or the detail of the probability segmenter were made because based on the separability of the pipeline we determined that the return in the form of test results on effort on that part of the system was low compared with other parts of the system. Though more subtle than the blatant overtraining used to detect the corner cases this is definitely a form of tuning to the test, overcoming it would require a test scored by a several distinct metrics which capture different aspects of system performance. If future projects of similar nature are done significant thought must be placed on the exact nature of the scoring to ensure broad coverage and to prevent tuning to the test.

Table 3 Final Results with corner cases

				No special rules (except for PT sheets)		**New rules added for corner cases**	
Target Type	Target Subtype	Level of Difficulty	Num Targets	Num Detected	PD [%]	Num Detected	PD [%]
Target	All	All	407	381	**93.6**	387	**95.1**
Target	Clay	All	111	107	96.4	107	96.4
Target	Rubber	All	158	150	94.9	151	95.6
Target	Saline	All	138	124	89.9	129	93.5
Target	Bulk	All	270	251	93	256	94.8
Target	Sheet	All	137	130	94.9	131	95.6
Target	All	Low	77	75	97.4	77	100
Target	Clay	Low	29	29	100	29	100
Target	Rubber	Low	22	22	100	22	100
Target	Saline	Low	26	24	92.3	26	100
Target	Bulk	Low	56	54	96.4	56	100
Target	Sheet	Low	21	21	100	21	100
Target	All	High	317	294	92.7	298	94
Target	Clay	High	82	78	95.1	78	95.1
Target	Rubber	High	125	118	94.4	119	95.2
Target	Saline	High	110	98	89.1	101	91.8
Target	Bulk	High	201	185	92	188	93.5
Target	Sheet	High	116	109	94	110	94.8
Pseudo-target	Sheet	High	10	10	**100**	10	**100**
			Num Non-targets	Num FAs	PFA [%]	Num FAs	PFA [%]
			1371	163	**11.9**	15	**1.1**
				Num Scans with FAs	Avg Num FAs	Num Scans with FAs	Avg Num FAs
				110	1.57	15	1

7 Summary

In this project we have developed an automated target recognition system that can detect targets from baggage X-ray data. Specific techniques include the use of a probability based segmenter based on features from 2D voxel slabs, the use of multiple independent segmenters, and random forest classifiers, we used these techniques explore the system performance through various stages of operation and explore what would be required to achieve perfect results.

8 References

ALERT. (2009). *Algorithm Development for Security Applications October 2009 Workshop Final Report.* Northeastern University, Boston, MA.

ALERT. (2014). *ALERT ATR Project: Top-Level Technical Specifications Version 5.*

Breiman, L. (2001). Random Forests. *Machine Learning 45(1)*, (pp. 5-32).

Crawford, C. R., Martz, H. E., & Pien, H. (n.d.). *Segmentation of Objects from.* Northeastern University, Boston, MA.

Eberhard, J. W., & Meng-Ling, H. (1998, Jan 27). *Patent No. 5712926.* US.

Hiraoglu, e. a. (2000, Jun 5). *Patent No. 6026179.* US.

Kim, H., Thiagarajan, J. J., & Bremer, T. (2014). Image Segmentation using Consensus from Hierarchical Segmentation Ensembles. *IEEE International Conference on Image Processing*, (pp. 3272-3276).

Kim, H., Thiagarajan, J. J., & Bremer, T. (2015). (Under Review) A Randomized Ensemble Approach to Industrial CT Segmentation. *Submitted to IEEE Computer Vision and Pattern Recognition (CVPR)*.

Kwon, J., Lee, J., & Song, S. M. (2013, Jul 4). *Patent No. 2013/0170723 A1.* US.

Megherbi, N., Flitton, G., & Breckon, T. (Sept 2010). A classifier based approach for the detection of potential threats in CT based Baggage Screening. *Image Processing (ICIP), 2010 17th IEEE International Conference on*, (pp. 1833-1836).

Murphy, K. P. (2012). *Machine Learning: A Probabalistic Perspective (Adaptive Computation and Machine learning Series).* MIT Press.

Simanovsky, S., Hiraoglu, M., Bechwati, I. M., & Crawford, C. (2000, Feb 15). *Patent No. 6111974.* US.

Woodhouse, D. F. (June 2012). Automatic Segmentation of CT scans of Checked Baggage. *Proceedings of the 2nd International Meeting on Image Formation in X-ray CT*, (pp. 310-313). Salt Lake City.

9 Acknowledgments

The authors would like to thank Timo Bremer who leads an LLNL-funded R&D project on CT reconstruction and segmentation. The ensemble segmenter reported on in this report was developed under Dr. Bremer's project. This report was excerpted from a conference paper submitted to the IEEE Computer Vision and Pattern Recognition (CVPR) 2015, which is now under review. The authors would like to acknowledge the

support and funding for this research by the Science & Technology Directorate of the Department of Homeland Security. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Appendix A Voxel classifier Evaluation

A.1 Random Forest Parameter Tuning

Random forests have only a small number of parameters to be tuned. We have explored the impact of two of these parameters, leaf size and number of trees, in the performance of voxel classifiers based on random forests.

Contrary to decision trees, in random forests, trees are not pruned. They are grown until pure terminal nodes are reached or until a user-defined leaf size is reached. Here the size of a leaf refers to the number of training examples that are assigned to that leaf. We have compared the performance of random forests with minimal terminal node size of one, four, ten, and fifty observations, while keeping all other input parameters equal. As seen on Figure A 1, the random forests were very robust to variations in the size of the leaves, as this parameter had no significant impact on their performance.

Similarly to the analysis performed regarding the impact on the size of leaves on performance of the random forests, we compared the performance of forests with 10,50, 100, 500, and 1000 trees, while keeping all other input parameters equal. Figure A 2 shows the performance of the random forests of different sizes. The smallest forests, with ten trees, performed worst, showing a wide range of AUC. Overall, starting at forests with 50 trees or more, the performances of the classifiers, as measured by AUC, were equivalent to one another and stable. However, when we compare the impact of the number of trees per classifier type (clay, saline, rubber, or powder) we see that especially for powder, increasing the number of trees from ten up to 500 in the random forest leads to incremental improvements in performance. For this reason, we used forests with 500 trees for the results shown in Section 3.1.2.2.

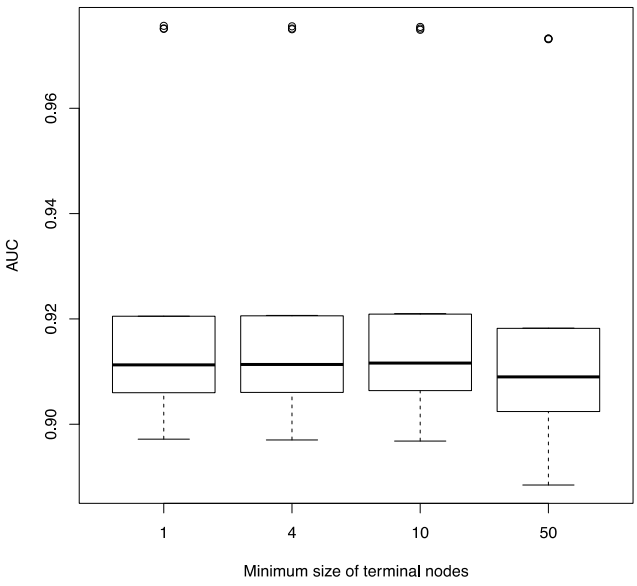


Figure A 1 Comparison of performance of random forest grown with different minimum size of terminal nodes

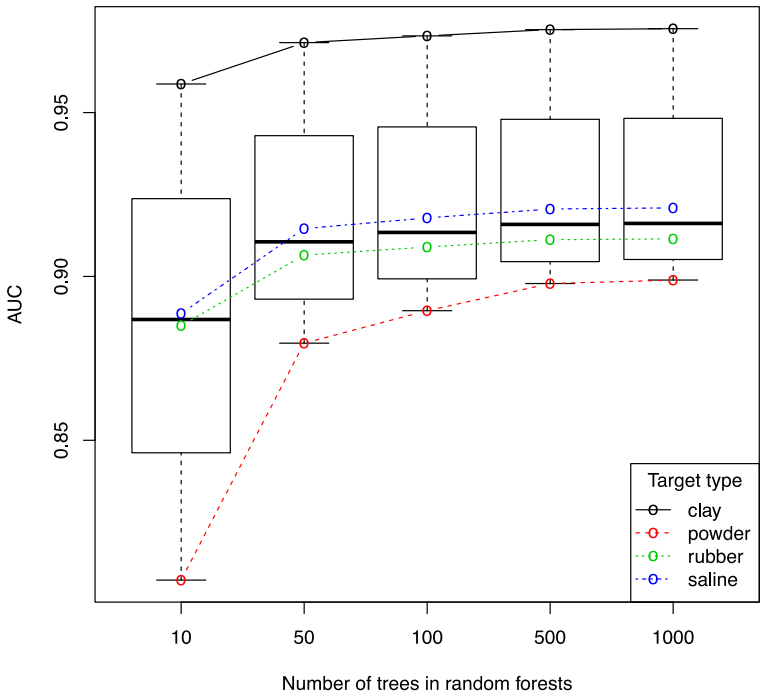


Figure A 2 Comparison of random forest performance based on number of trees.

A.2 Training Set Type

The bag data provided is unbalanced, that is, the number of voxels associated with non-targets is much larger than the number of voxels associated with targets. In particular, the percentage of voxels assigned to the categories non-target, saline, rubber, powder, and clay are respectively 79.1%, 5.78%, 9.12%, 0.93%, and 5.03%.

Training classifiers with unbalanced training sets can lead to biased classifiers in the sense that the majority class will be emphasized over the minority class. One approach to overcome this issue is to down-sample the majority class observations (in this case, non-target voxels) in the training data in order to obtain a balanced, or close-to-balanced, training set, which is a training set in which the number of positive and negative examples are approximately the same. We have built random forests using both balanced and unbalanced training set to assess how that affects the performance of the voxel classifiers.

For the unbalanced set, we used approximately 10% of the non-target voxels, but the training sets were still highly unbalanced. Table 3 shows the total number of positive and negative voxels in the training set for each type of classifier. For a classifier of a given type, only voxels of that type are considered positive voxels, and non-target voxels as well as the other three target types are treated as negative voxels.

The balanced training sets for each classifier were built to meet two constraints: 1) The number of positive and negative voxels must be approximately the same; and 2) The training set size must be as large as possible, being limited only by the number of positive voxels and the need to perform three-fold cross-validation. The negative observations of each type (non-targets, and the three remaining target types) were reduced via random sampling. This set containing all positive voxels and a sub-sample of the negative voxels was then subjected to random sampling to compose the three training sets for the three-fold cross-validation sets. Table A 1 shows the number of observations in the balanced training set of the four types of targets.

Figure A 3 shows the AUC of random forests built for each one of the four types of targets (saline, rubber, powder, and clay) when using balanced versus unbalanced training sets. Using balanced training sets consistently improved the AUC of the random forests.

Table A 1 Number of positive and negative voxel slabs in the unbalanced training sets for each of the four types of targets.

	Positive	**Negative**
Saline	28633	134039
Rubber	45439	117233
Powder	4584	158088
Clay	25073	137599

Table A 2 Number of positive and negative voxel slabs in the balanced training sets for each of the four types of targets.

	Positive	**Negative**
Saline	28778	26170
Rubber	45310	38383
Powder	4623	4516
Clay	24985	23610

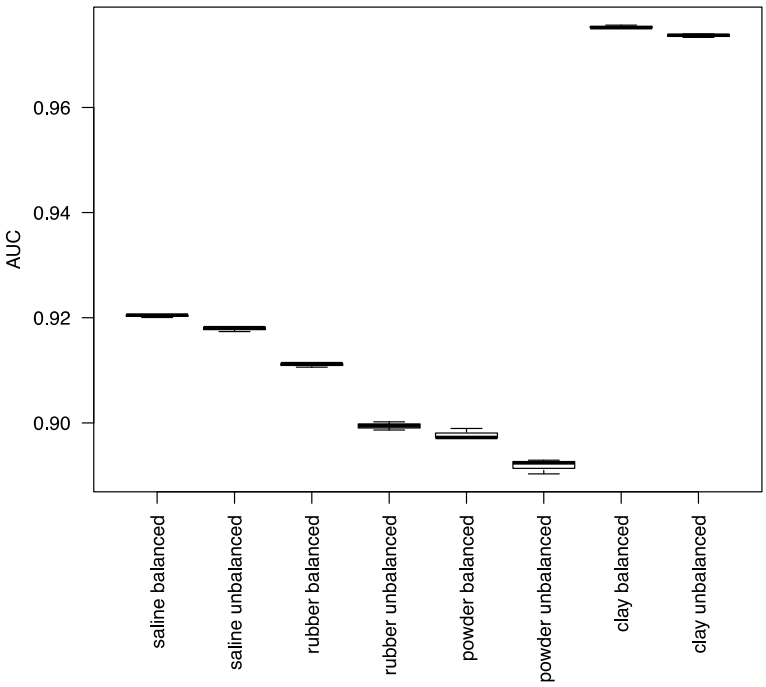


Figure A 3 Performance of random forests based on type of training set: balanced versus unbalanced.

A.3 Classification based on object ID

In an effort to achieve the best voxel classification results, we have explored the performance of random forests trained on a per-object basis. The rationale behind building classifiers per object is that this could potentially be an approximation of an upper bound of how well type classifiers could do, recognizing that any actual classifier designed in this fashion would likely be over-trained. The idea is that the variance in the voxels properties within each object should be no greater than the variance of the same voxel properties for voxels of a given type, regardless of the object ID. If this is the case, then object ID classifiers should yield better prediction accuracy than the type classifiers, whose results we have described in Section 3.1.2.1. Comparing the between and within object ID variance in some of the important predictor variables (such as voxel slab median) also suggests that building object ID classifiers may lead to improved results (see Table 5).

Hence, we have trained 93 different classifiers, one for each object ID (The number of object IDs of types saline, rubber, powder, and clay are respectively 29, 30, 7, and 27). Figure 11.A shows that there is a modest increase in the median performance of classifiers for all types, with the exception of clay. Interestingly, the variance in the performance of the classifiers has increased considerably in contrast to the performance of the type classifiers (compare Figures 6 and A 4.A). A large part of this increase in variance of performance is due to the variable size of each object. Overall, classifiers trained on larger objects (a.k.a., objects with a larger number of voxel slabs) tend to yield better performance accuracy, as measured by AUC. Figure A 4.B shows the dependency

between classifier performance and number of positive instances in the training set (represented by the number of voxels of the corresponding object).

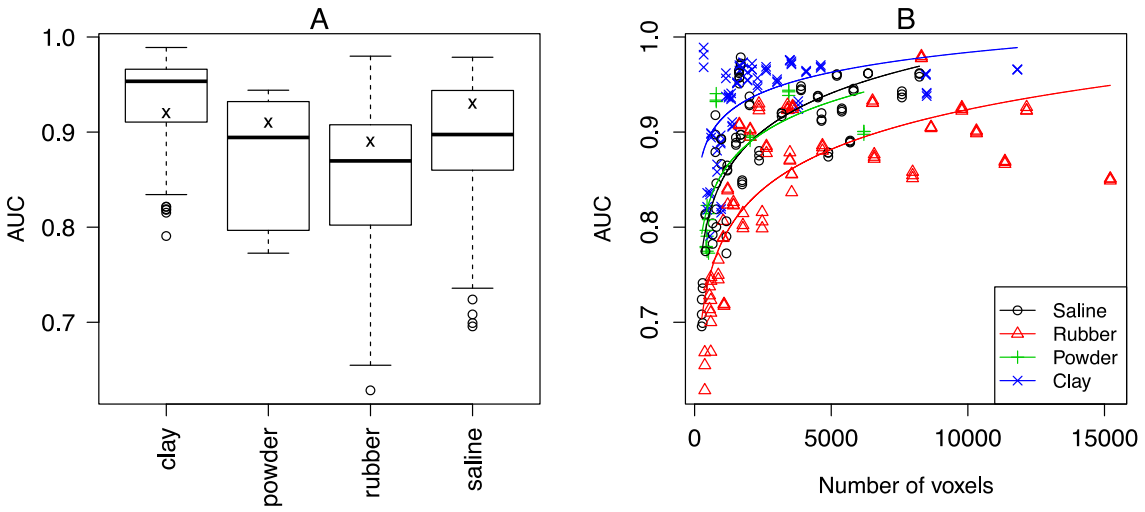


Figure A 4 (A) AUCs obtained by classifiers for the four object subtypes: saline, rubber, powder, and clay. Each box represents the results from three-fold cross-validation for each classifier of objects belonging to the corresponding target type (Number of classifiers of clay, powder, rubber, and saline were respectively 27, 7, 30, and 29). The "x" shows the median AUC of the three-fold cross-validation of the type classifiers shown in Figure 6. (B) AUC of the individual classifiers colored by the target type of each object ID. The solid lines show the regression of AUC as a function of the logarithm of the number of voxels. The R^2 of the regression for saline, rubber, powder, and clay were 0.61, 0.61, 0.53, and 0.31, respectively.

Table A 3 Anova model "Median ~ Type + ObjectId". Significant F values indicate that ObjectId contributes to the variance in the "Median" variable even after "Type" has been accounted for.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Type	4	42722180843.98	10680545211.00	23067.82	0.0000
ObjectId	1	54542529.44	54542529.44	117.80	0.0000
Residuals	1492488	691031345655.29	463006.30		

A.4 Other Algorithms Considered for Voxel Classification

We considered a variety of machine learning algorithms as candidates for our voxel classifier, including random forests, support vector machines (SVM), adaptive boosting, nearest neighbors (NN), and naïve Bayes (NB)[3]. Overall, the best performances were obtained using variations of random forests, which was the primary reason for choosing this method for inclusion in our pipeline (further, secondary reasons for choosing random

[3] For details on these algorithms, see (Murphy, 2012).

forests over the other algorithms explored are described in Section 3.1.2.1). As explained above, the goal of the voxel classifier is to produce a filtered set that is a subset of the original data enriched for target voxels (that is, having as many as possible non-target voxels removes while retaining as many as possible target voxels). Table A 4 shows a direct comparison of the composition of filtered sets obtained by each of the five algorithms explored, both in terms of total percentage of voxels, as well as in terms of the percentage of target and non-target voxels. Naïve Bayes and SVMs tended to classify most voxels as "positive", only removing a small proportion of voxels from the filtered set, which was insufficient to adequately aid in the connect component segmentation. Nearest neighbors and random forests were the most successful approaches in reducing the size of the filtered set (both reducing the data to about 60% of the original number of voxels), with nearest neighbors actually being slightly more successful than random forests in this aspect. However, on a per-object basis, random forests were able to retain the largest fraction of positive voxels. Table A 5shows the percentage of voxels of each type (nothing, saline, rubber, powder, and clay) from the original dataset included in the filtered dataset.

Table A 4 : Summary statistics about filtered sets obtained via different algorithms. * value excludes object ID 7011, for which fewer voxels were detected.

	% total voxels	% non-target voxels	min % target voxels	min % voxels per obj
Random Forest	61.8	55	96	84*
Boosting	64.3	58	95	86*
Naive Bayes	92.2	91	98	86
SVM	87.4	85	97	90*
NN	60	53	91	80*

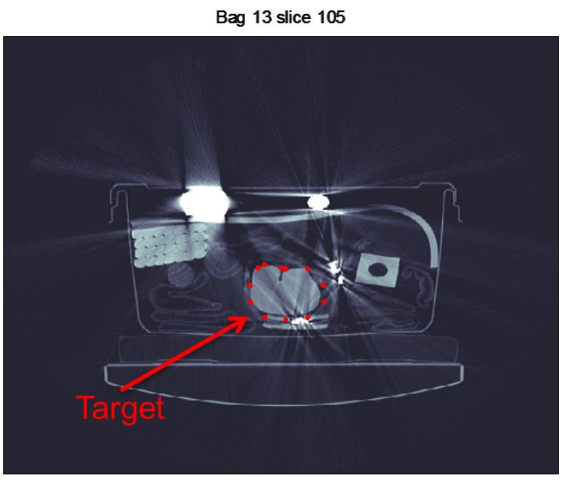
Table A 5 Percentage of voxels of each type (nothing, saline, rubber, powder, clay) from the cleaned dataset included in filtered dataset based on classifier type, using a threshold to obtain voxel TPR >= 0.9. NN_b = nearest neighbor algorithm using a balanced training set (i.e., a training set with equal size classes).

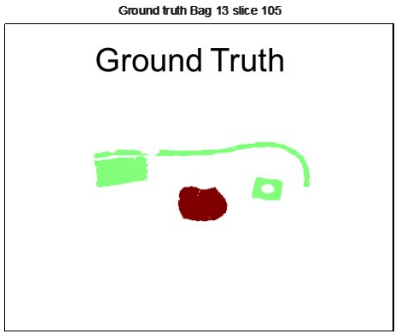
	nothing	saline	rubber	powder	clay
Random Forest	0.55	0.98	0.97	0.96	0.98
Boosting	0.58	0.97	0.97	0.95	0.98
Naive Bayes	0.91	0.98	0.99	0.99	0.99
SVM	0.85	0.99	0.99	0.97	0.99
NN	0.53	0.98	0.97	0.91	0.98
NN_b	0.63	0.98	0.98	0.97	0.99

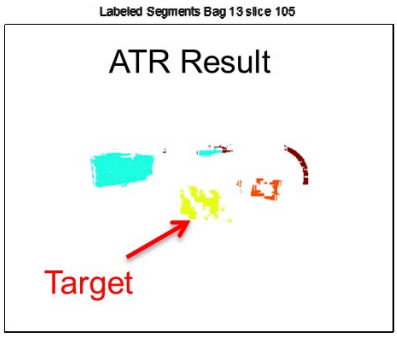
Appendix B Example Results

We show here a number of example cases of the results generated with the described pipeline along with some commentary on the results. All the cases show 3 images including a view of the image data. The same slice of the labeled ground truth data and another image of the labeled image data output from the ATR pipeline. The original image data is shown in grayscale with the white being the highest density. The upper limit on the image color scale is 2500. The target itself is surrounded by red dots. The ground truth data is shown with a white background the object in question is shown in red and other labeled targets are shown in green. The labeled image data is shown with a different color for each individual label. The colors appear in somewhat random fashion. For each of the cases examined, the actual precision and recall is shown along with a brief discussion about the situation and the effect on the results.

Case #1: Bulk with bad streaks caused by metal



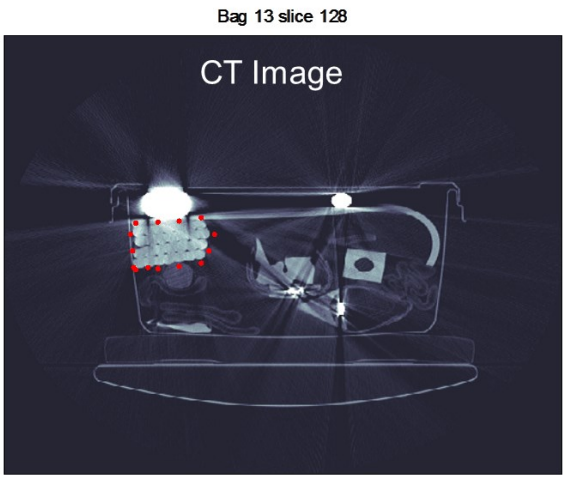


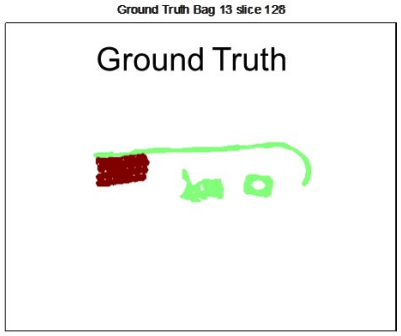


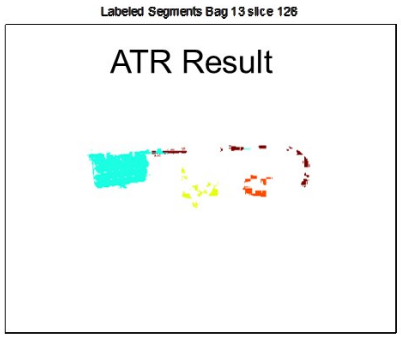
Detected: YES
Precision: 95.2%
Recall: 60.1%

Due to the streaking artifacts some parts of the target object were removed in the final image, thus the recall is reduced.

Case #2: Bulk with bad shading caused by beam hardening and scatter



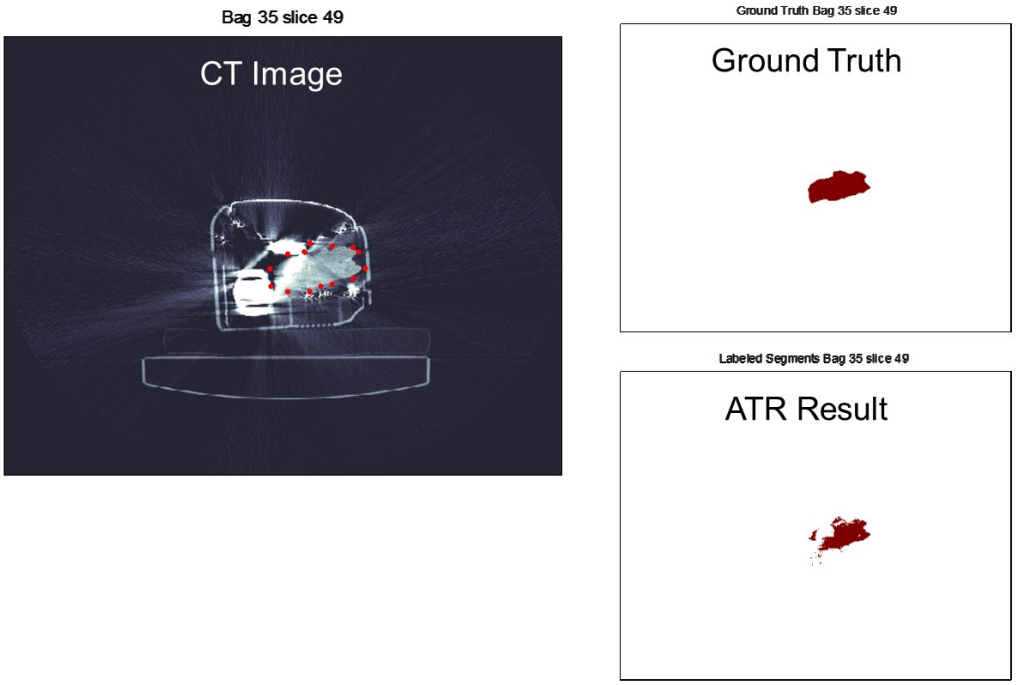




Detected: YES
Precision: 72.2%
Recall: 94.2%

The target is merged with the overlapping portion of the sheet on top of the target reducing the precision

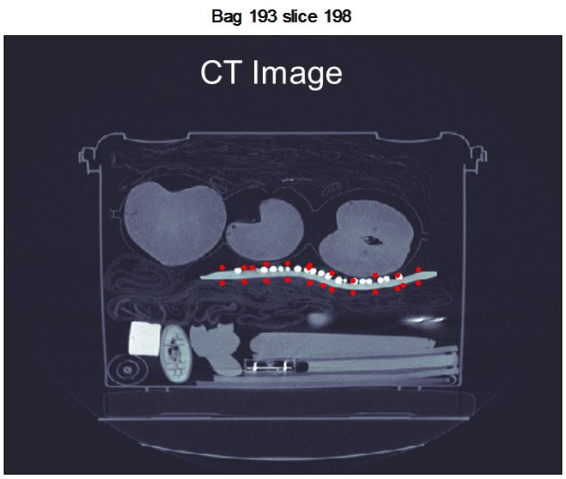
Case #3: Bulk inside electronics

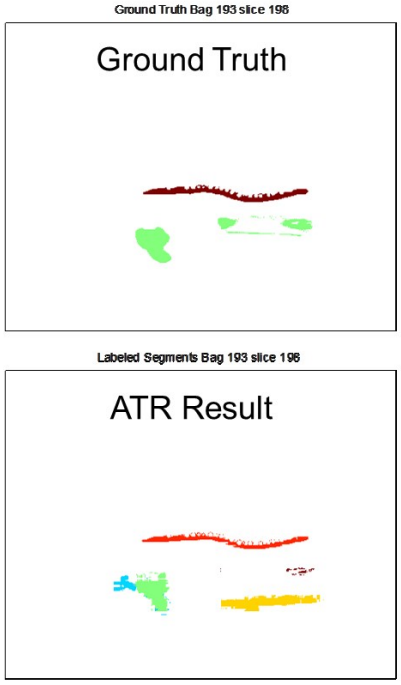


Detected: YES
Precision: 87.5%
Recall: 79.3%

Some parts of the target image are removed due to the metal artifacts

Case #4: Bulk with texture

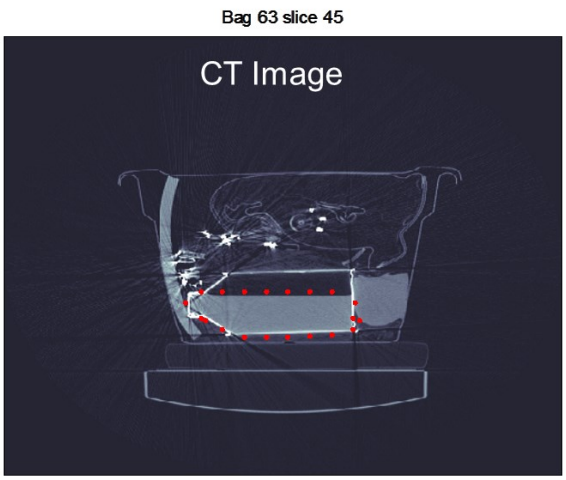


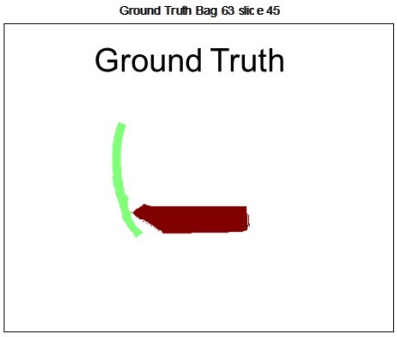


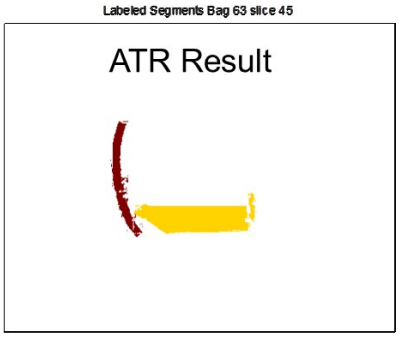
Detected: YES
Precision: 96.5%
Recall: 73.8%

The ground truth label identifies some of the texture as part of the target. The ATR removes most of them resulting in a lower recall

Case #5: Bulk with density close to water (~5% saline)



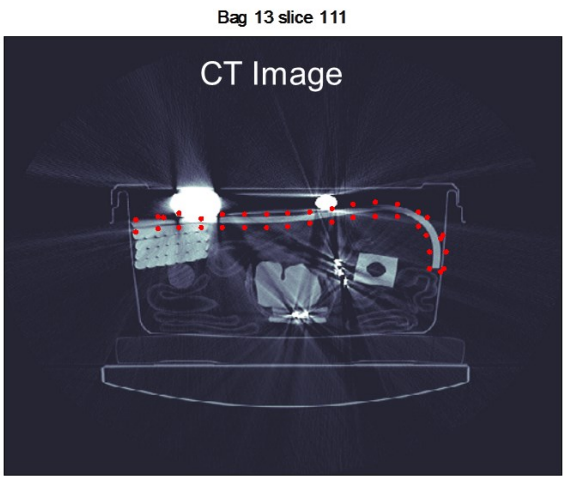


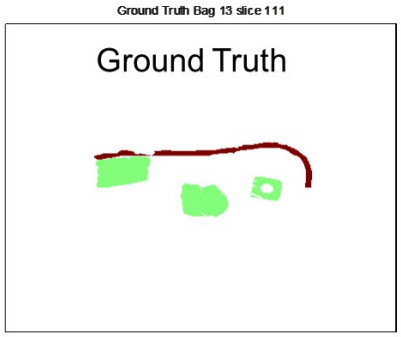


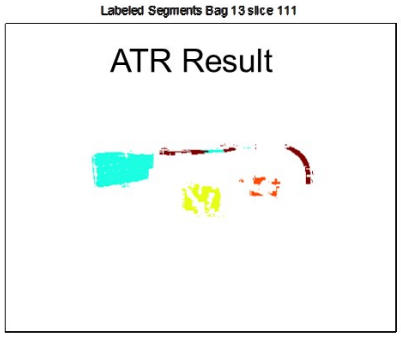
Detected: YES
Precision: 93%
Recall: 95.5%

Object was detected with no issues

Case #6: Sheet with bad streaks caused by metal, beam hardening and scatter



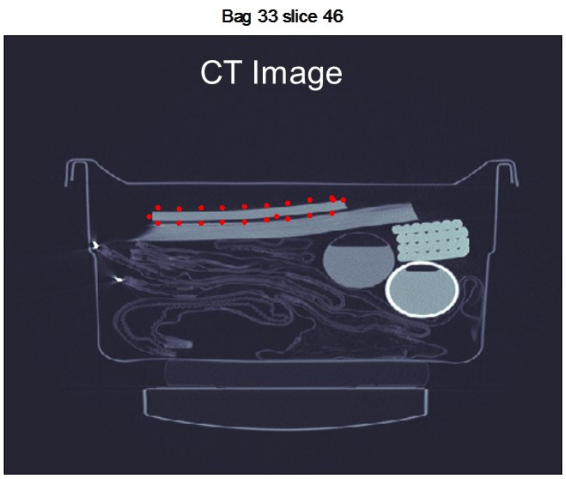


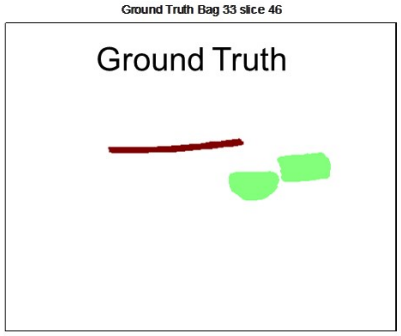


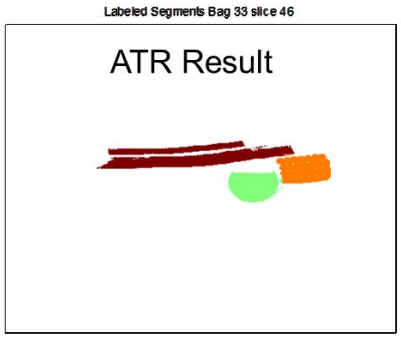
Detected: YES
Precision: 83.3%
Recall: 26.7%

The sheet was split into multiple pieces resulting in a low recall.

Case #7: Sheet laying on top of another flat object



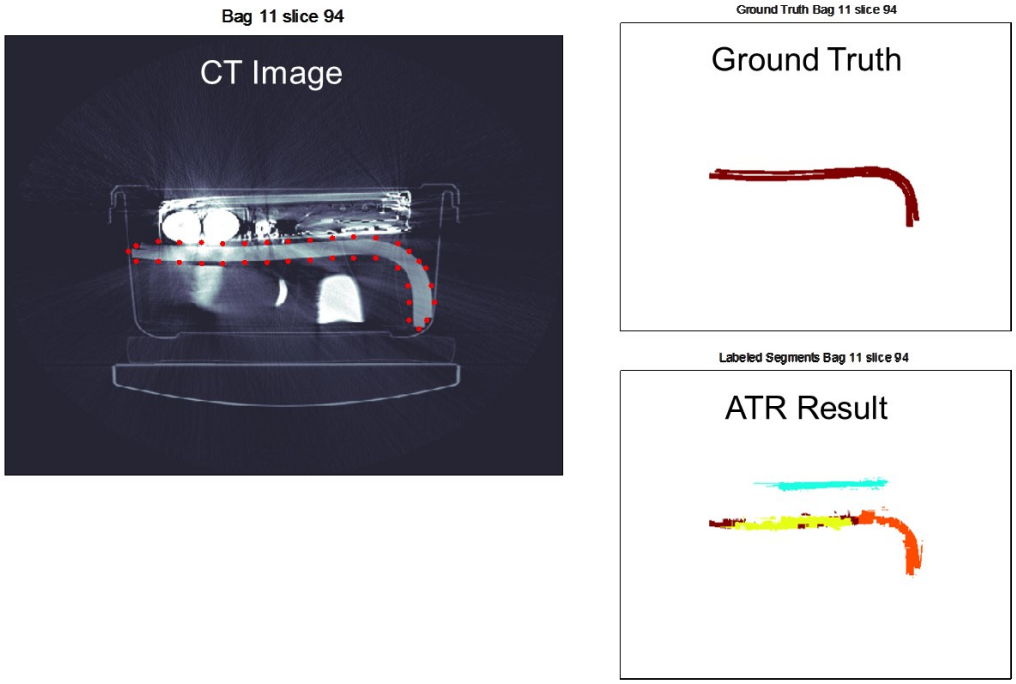




Detected: YES
Precision: 21.1%
Recall: 82.7%

The sheet is detected fully but is merged with the object below it, though still met the test criteria. In this plane the objects are separated but in other planes they appear merged. A more sophisticated splitting algorithm based on image voxel data instead of voxel slabs probably would have separated the two objects, but such an algorithm was not necessary to meet the requirements of this test.

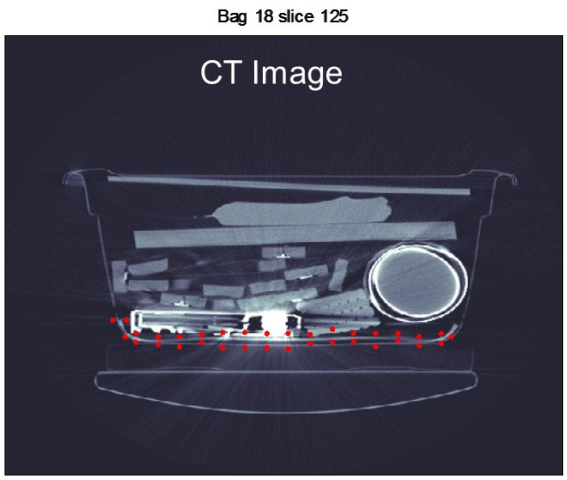
Case #8: Object with lots of photon starvation

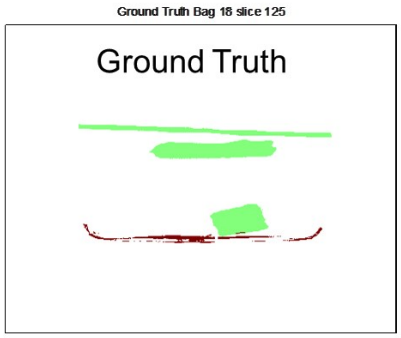


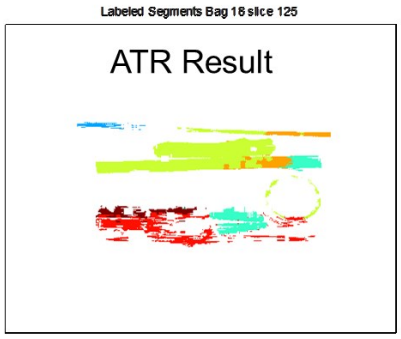
Detected: YES
Precision: 71.9%
Recall: 44.4%

The sheet is split into multiple objects reducing the recall of the object. Improved merging algorithms would probably be capable of merging the object but were not necessary for this test. Note also there is a false alarm object in this frame.

Case #9: PT sheet based on thickness



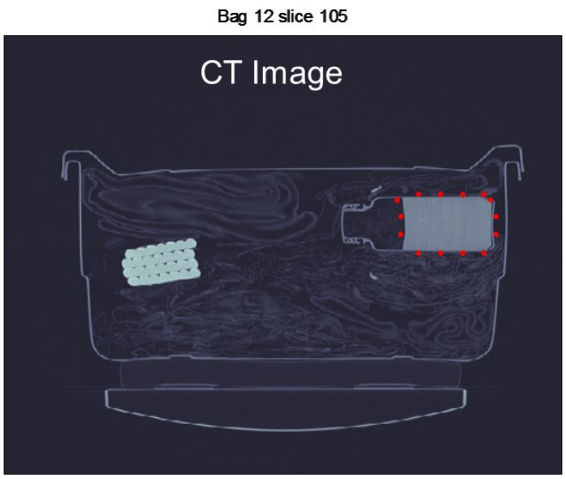


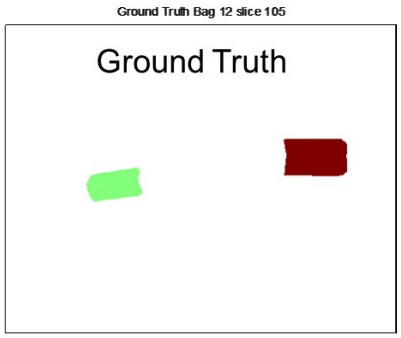


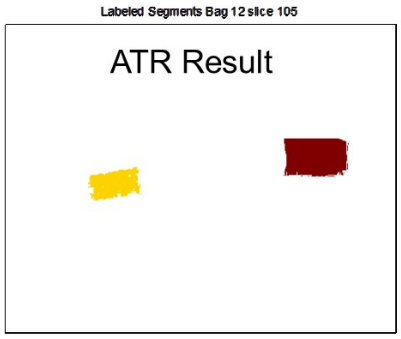
Detected: YES
Precision: 23.2%
Recall: 32.6%

This sheet is not particularly well captured but sufficiently to count as a detection in this test. The sheet itself appears merged with some of its surrounding objects.

Case #10: PT Powder (based on density, not mass)



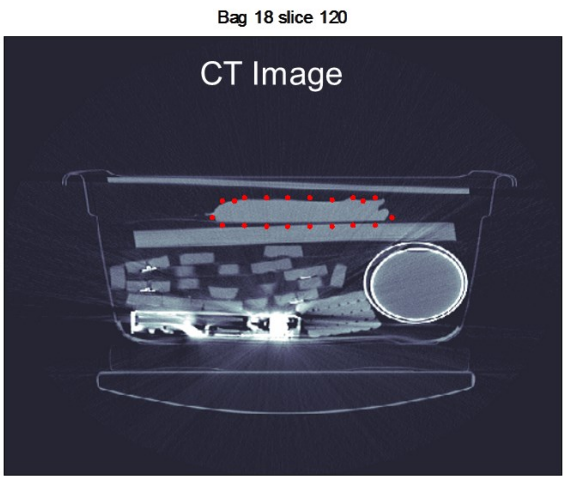


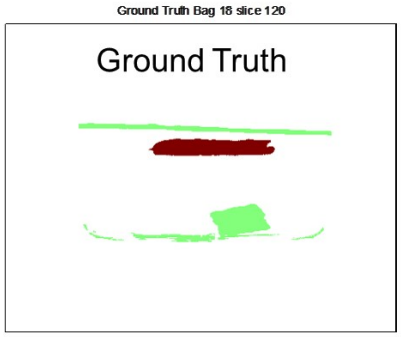


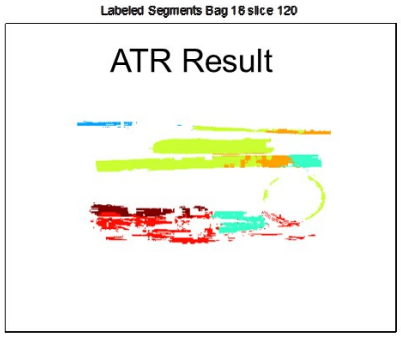
Detected: NO
Precision: 49.95%
Recall: 96%

Powders were not considered detection requirements in the final version of the test so the powder detector was mostly turned off, though detection did not count against the final score so some were detected. In this case the object is merged with another object of similar size and shape behind the frame in this image so it is not visible.

Missed Detection #1: Merger



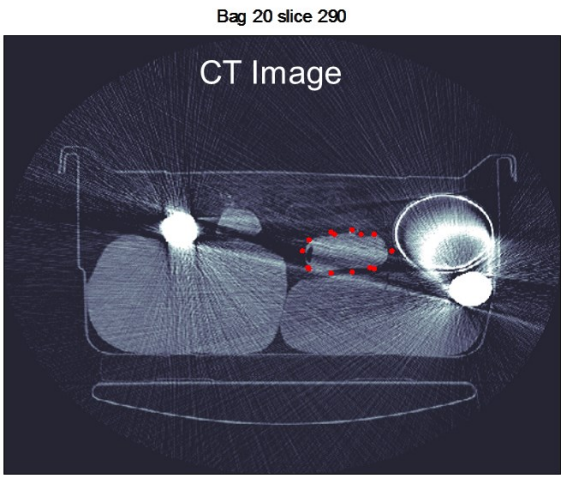


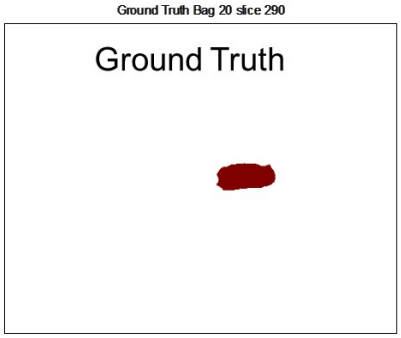


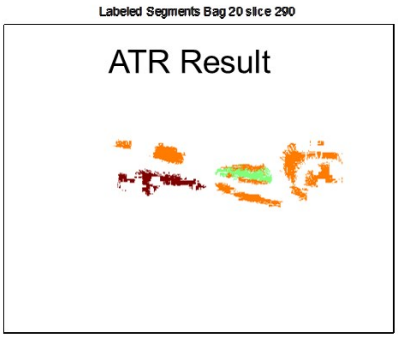
Detected: NO
Precision: 28.1%
Recall: 90.4%

The object in question is merged with a non-target object of similar density below it in the image.

Missed Detection #2: Metal artifacts







Detected: NO
Precision: 23.0%
Recall: 38.4%

The object in question is badly distorted by metal artifacts so the object itself was split into multiple pieces with different apparent densities, and the ATR did not piece enough of it together to count as a detection.

There were a total of 13 missed targets after the split and merge stage the exact specifications are shown in following table

Bag	Target	type	Prec	recall
13	6047	R	92	47
15*	6045	C	98	46
16*	6002	S	33	97
18*	6025	S	28	90
18*	6051	C	79	32
18*	8031	R sh	5	17
20	6012	S	23	38
34	6012	S	95	43
38	6001	S	41	98
115	6178	S	46	92
147*	6140	R sh	18	65
162*	6573	R sh	15	93
183	6557	S	20	65

- objects were detected in some tests but not final results

4 missed detections due to splitting
7 missed detections due to merging
2 missed detections had both split and merge issues